CUDA Debugger API

API Reference Manual
# Table of Contents

Chapter 1. Release Notes.................................................................................................... 1
  1.1. 11.8 Release.................................................................................................................. 1
  1.2. 7.0 Release................................................................................................................... 1
  1.3. 6.5 Release................................................................................................................... 2

Chapter 2. Introduction.................................................................................................... 3
  2.1. Debugger API.................................................................................................................. 3
  2.2. ELF and DWARF............................................................................................................. 4
  2.3. ABI Support.................................................................................................................... 6
  2.4. Exception Reporting....................................................................................................... 6
  2.5. Attaching and Detaching............................................................................................... 6

Chapter 3. Modules......................................................................................................... 9
  3.1. General........................................................................................................................... 9
    CUDBGResult.................................................................................................................... 9
  3.2. Initialization.................................................................................................................. 11
    CUDBGAPI_st::finalize...................................................................................................... 12
    CUDBGAPI_st::initialize................................................................................................. 12
  3.3. Device Execution Control............................................................................................. 12
    CUDBGAPI_st::resumeDevice.......................................................................................... 12
    CUDBGAPI_st::resumeWarpsUntilPC.............................................................................. 13
    CUDBGAPI_st::singleStepWarp...................................................................................... 14
    CUDBGAPI_st::singleStepWarp40.................................................................................. 14
    CUDBGAPI_st::singleStepWarp41.................................................................................. 15
    CUDBGAPI_st::suspendDevice....................................................................................... 16
  3.4. Breakpoints.................................................................................................................. 16
    CUDBGAPI_st::getAdjustedCodeAddress....................................................................... 16
    CUDBGAPI_st::setBreakpoint........................................................................................ 17
    CUDBGAPI_st::setBreakpoint31.................................................................................... 18
    CUDBGAPI_st::unsetBreakpoint..................................................................................... 18
    CUDBGAPI_st::unsetBreakpoint31............................................................................... 19
  3.5. Device State Inspection............................................................................................... 19
    CUDBGAPI_st::getLoadedFunctionInfo.......................................................................... 19
    CUDBGAPI_st::getManagedMemoryRegionInfo............................................................. 20
    CUDBGAPI_st::memcheckReadErrorAddress................................................................... 21
    CUDBGAPI_st::readActiveLanes..................................................................................... 21
    CUDBGAPI_st::readBlockIdx......................................................................................... 22
CUDBGAPI_st::readBlockIdx32.................................................................................................. 23
CUDBGAPI_st::readBrokenWarps............................................................................................. 24
CUDBGAPI_st::readCallDepth............................................................................................... 25
CUDBGAPI_st::readCallDepth32............................................................................................ 25
CUDBGAPI_st::readCCRegister............................................................................................... 26
CUDBGAPI_st::readClusterIdx............................................................................................. 27
CUDBGAPI_st::readCodeMemory............................................................................................ 28
CUDBGAPI_st::readConstMemory............................................................................................ 29
CUDBGAPI_st::readErrorPC.................................................................................................... 30
CUDBGAPI_st::readGenericMemory......................................................................................... 30
CUDBGAPI_st::readGlobalMemory.......................................................................................... 31
CUDBGAPI_st::readGlobalMemory31....................................................................................... 32
CUDBGAPI_st::readGlobalMemory55....................................................................................... 33
CUDBGAPI_st::readGridId...................................................................................................... 34
CUDBGAPI_st::readGridId50................................................................................................... 35
CUDBGAPI_st::readLaneException......................................................................................... 36
CUDBGAPI_st::readLaneStatus............................................................................................... 36
CUDBGAPI_st::readLocalMemory........................................................................................... 37
CUDBGAPI_st::readParamMemory.......................................................................................... 38
CUDBGAPI_st::readPC............................................................................................................ 39
CUDBGAPI_st::readPinnedMemory......................................................................................... 40
CUDBGAPI_st::readPredicates............................................................................................... 41
CUDBGAPI_st::readRegister................................................................................................. 42
CUDBGAPI_st::readRegisterRange......................................................................................... 43
CUDBGAPI_st::readReturnAddress...................................................................................... 44
CUDBGAPI_st::readReturnAddress32.................................................................................... 45
CUDBGAPI_st::readSharedMemory......................................................................................... 46
CUDBGAPI_st::readSyscallCallDepth..................................................................................... 47
CUDBGAPI_st::readTextureMemory....................................................................................... 47
CUDBGAPI_st::readTextureMemoryBindless.......................................................................... 48
CUDBGAPI_st::readThreadIdx............................................................................................... 49
CUDBGAPI_st::readUniformPredicates................................................................................... 50
CUDBGAPI_st::readUniformRegisterRange............................................................................. 51
CUDBGAPI_st::readValidLanes.............................................................................................. 52
CUDBGAPI_st::readValidWarps............................................................................................. 52
CUDBGAPI_st::readVirtualPC............................................................................................... 53
CUDBGAPI_st::readVirtualReturnAddress............................................................................. 54
CUDBGAPI_st::readVirtualReturnAddress32......................................................................... 55
3.6. Device State Alteration

CUDBGAPI_st::readWarpState
CUDBGAPI_st::readWarpState60
CUDBGAPI_st::writePinnedMemory
CUDBGAPI_st::writePredicates
CUDBGAPI_st::writeUniformPredicates

3.7. Grid Properties

CUDBGGridInfo
CUDBGGridStatus
CUDBGAPI_st::getBlockDim
CUDBGAPI_st::getClusterDim
CUDBGAPI_st::getElfImage
CUDBGAPI_st::getElfImage32
CUDBGAPI_st::getGridAttribute
CUDBGAPI_st::getGridAttributes
CUDBGAPI_st::getGridDim
CUDBGAPI_st::getGridDim32
CUDBGAPI_st::getGridInfo
CUDBGAPI_st::getGridInfo55
CUDBGAPI_st::getGridStatus
CUDBGAPI_st::getGridStatus50
CUDBGAPI_st::getTID

3.8. Device Properties

CUDBGAPI_st::getDeviceName
CUDBGAPI_st::getDeviceType
CUDBGAPI_st::getNumDevices
CUDBGAPI_st::getNumLanes
CUDBGAPI_st::getNumPredicates
3.9. DWARF Utilities .......................................................... 83
CUDBGAPI_st::disassemble .............................................. 83
CUDBGAPI_st::getElfImageByHandle ......................... 84
CUDBGAPI_st::getHostAddrFromDeviceAddr .............. 84
CUDBGAPI_st::getPhysicalRegister30 ...................... 85
CUDBGAPI_st::getPhysicalRegister40 ...................... 86
CUDBGAPI_st::isDeviceCodeAddress ....................... 87
CUDBGAPI_st::isDeviceCodeAddress55 ..................... 87
CUDBGAPI_st::lookupDeviceCodeSymbol .................. 88

3.10. Events ..................................................................... 88
CUDBGEvent .................................................................. 89
CUDBGEventCallbackData ........................................... 89
CUDBGEventCallbackData40 ......................................... 89
CUDBGEventKind ....................................................... 89
CUDBGNotifyNewEventCallback ................................. 90
CUDBGNotifyNewEventCallback31 ............................. 90
CUDBGAPI_st::acknowledgeEvent30 ......................... 90
CUDBGAPI_st::acknowledgeEvents42 ......................... 91
CUDBGAPI_st::acknowledgeSyncEvents ..................... 91
CUDBGAPI_st::getNextAsyncEvent50 ......................... 91
CUDBGAPI_st::getNextAsyncEvent55 ......................... 92
CUDBGAPI_st::getNextEvent ....................................... 92
CUDBGAPI_st::getNextEvent30 ................................... 92
CUDBGAPI_st::getNextEvent32 ................................... 93
CUDBGAPI_st::getNextEvent42 ................................... 93
CUDBGAPI_st::getNextSyncEvent50 ......................... 94
CUDBGAPI_st::getNextSyncEvent55 ......................... 94
CUDBGAPI_st::setNotifyNewEventCallback ............... 94
CUDBGAPI_st::setNotifyNewEventCallback31 .......... 95
CUDBGAPI_st::setNotifyNewEventCallback40 .......... 95

Chapter 4. Data Structures ............................................. 97
acknowledgeEvent30................................................................. 98
acknowledgeEvents42.............................................................. 98
acknowledgeSyncEvents.......................................................... 98
clearAttachState........................................................................ 98
disable....................................................................................... 98
finalize...................................................................................... 99
getAdjustedCodeAddress.......................................................... 100
getBlockDim.............................................................................. 100
getClusterDim.......................................................................... 101
getDeviceName.......................................................................... 101
getDevicePCIBusInfo................................................................. 102
getDeviceType............................................................................ 102
gElfimage..................................................................................... 103
gElfimage32................................................................................. 104
gElfimageByHandle..................................................................... 104
getGridAttribute......................................................................... 105
getGridAttributes....................................................................... 106
gGetGridDim.............................................................................. 106
gGetGridDim32........................................................................... 107
gGetGridInfo............................................................................... 108
gGetGridInfo55........................................................................... 108
gGetGridStatus........................................................................... 109
gGetGridStatus50...................................................................... 109
gHostAddrFromDeviceAddr........................................................... 110
gGetLoadedFunctionInfo............................................................... 110
getManagedMemoryRegionInfo.................................................... 111
gNextAsyncEvent50...................................................................... 112
gNextAsyncEvent55..................................................................... 112
gNextEvent............................................................................... 112
gNextEvent30.............................................................................. 113
gNextEvent32.............................................................................. 113
gNextEvent42.............................................................................. 114
gNextSyncEvent50...................................................................... 114
gNextSyncEvent55...................................................................... 114
gGetNumDevices.......................................................................... 114
gGetNumLanes............................................................................. 115
gGetNumPredicates..................................................................... 116
gGetNumRegisters....................................................................... 116
readPinnedMemory................................................................................................................... 144
readPredicates.......................................................................................................................... 145
readRegister.............................................................................................................................. 146
readRegisterRange................................................................................................................... 147
readReturnAddress................................................................................................................... 148
readReturnAddress32................................................................................................................. 148
readSharedMemory.................................................................................................................. 149
readSyscallCallDepth................................................................................................................. 150
readTextureMemory.................................................................................................................. 151
readTextureMemoryBindless....................................................................................................... 152
readThreadIdx........................................................................................................................... 153
readUniformPredicates............................................................................................................. 154
readUniformRegisterRange....................................................................................................... 154
readValidLanes.......................................................................................................................... 155
readValidWarps......................................................................................................................... 156
readVirtualPC............................................................................................................................ 157
readVirtualReturnAddress......................................................................................................... 157
readVirtualReturnAddress32..................................................................................................... 158
readWarpState.......................................................................................................................... 159
readWarpState60....................................................................................................................... 160
requestCleanupOnDetach........................................................................................................ 160
requestCleanupOnDetach55....................................................................................................... 161
resumeDevice............................................................................................................................ 161
resumeWarpsUntilPC............................................................................................................... 161
setBreakpoint............................................................................................................................ 162
setBreakpoint31........................................................................................................................ 163
setKernelLaunchNotificationMode.......................................................................................... 163
setNotifyNewEventCallback..................................................................................................... 163
setNotifyNewEventCallback31................................................................................................. 164
setNotifyNewEventCallback40................................................................................................. 164
singleStepWarp......................................................................................................................... 165
singleStepWarp40..................................................................................................................... 165
singleStepWarp41..................................................................................................................... 166
suspendDevice.......................................................................................................................... 167
unsetBreakpoint....................................................................................................................... 167
unsetBreakpoint31.................................................................................................................... 168
writeCCRegister........................................................................................................................ 168
writeGenericMemory................................................................................................................. 169
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>writeGlobalMemory</td>
<td>170</td>
</tr>
<tr>
<td>writeGlobalMemory31</td>
<td>171</td>
</tr>
<tr>
<td>writeGlobalMemory55</td>
<td>171</td>
</tr>
<tr>
<td>writeLocalMemory</td>
<td>172</td>
</tr>
<tr>
<td>writeParamMemory</td>
<td>173</td>
</tr>
<tr>
<td>writePinnedMemory</td>
<td>174</td>
</tr>
<tr>
<td>writePredicates</td>
<td>175</td>
</tr>
<tr>
<td>writeRegister</td>
<td>176</td>
</tr>
<tr>
<td>writeSharedMemory</td>
<td>177</td>
</tr>
<tr>
<td>writeUniformPredicates</td>
<td>178</td>
</tr>
<tr>
<td>writeUniformRegister</td>
<td>178</td>
</tr>
<tr>
<td>CUDBGEvent</td>
<td>179</td>
</tr>
<tr>
<td>cases</td>
<td>179</td>
</tr>
<tr>
<td>kind</td>
<td>179</td>
</tr>
<tr>
<td>CUDBGEvent::cases_st</td>
<td>179</td>
</tr>
<tr>
<td>contextCreate</td>
<td>180</td>
</tr>
<tr>
<td>contextDestroy</td>
<td>180</td>
</tr>
<tr>
<td>contextPop</td>
<td>180</td>
</tr>
<tr>
<td>contextPush</td>
<td>180</td>
</tr>
<tr>
<td>elfImageLoaded</td>
<td>180</td>
</tr>
<tr>
<td>internalError</td>
<td>180</td>
</tr>
<tr>
<td>kernelFinished</td>
<td>180</td>
</tr>
<tr>
<td>kernelReady</td>
<td>180</td>
</tr>
<tr>
<td>CUDBGEvent::cases_st::contextCreate_st</td>
<td>181</td>
</tr>
<tr>
<td>context</td>
<td>181</td>
</tr>
<tr>
<td>dev</td>
<td>181</td>
</tr>
<tr>
<td>tid</td>
<td>181</td>
</tr>
<tr>
<td>CUDBGEvent::cases_st::contextDestroy_st</td>
<td>181</td>
</tr>
<tr>
<td>context</td>
<td>182</td>
</tr>
<tr>
<td>dev</td>
<td>182</td>
</tr>
<tr>
<td>tid</td>
<td>182</td>
</tr>
<tr>
<td>CUDBGEvent::cases_st::contextPop_st</td>
<td>182</td>
</tr>
<tr>
<td>context</td>
<td>182</td>
</tr>
<tr>
<td>dev</td>
<td>182</td>
</tr>
<tr>
<td>tid</td>
<td>182</td>
</tr>
<tr>
<td>CUDBGEvent::cases_st::contextPush_st</td>
<td>182</td>
</tr>
<tr>
<td>context</td>
<td>183</td>
</tr>
<tr>
<td>dev</td>
<td>183</td>
</tr>
</tbody>
</table>
tid ............................................................................................................................................... 183
CUDBGEvent::cases_st::elfImageLoaded_st .............................................................................. 183
context ....................................................................................................................................... 184
dev ..............................................................................................................................................184
handle ........................................................................................................................................ 184
module .......................................................................................................................................184
properties .................................................................................................................................. 184
size .............................................................................................................................................184
CUDBGEvent::cases_st::internalError_st ................................................................................... 184
cntorType ................................................................................................................................... 185
CUDBGEvent::cases_st::kernelFinished_st ................................................................................ 185
cntor ........................................................................................................................................... 186
dev ..............................................................................................................................................186
function ..................................................................................................................................... 186
functionEntry ............................................................................................................................. 186
gridId ..........................................................................................................................................186
module .......................................................................................................................................186
tid ............................................................................................................................................... 186
CUDBGEvent::cases_st::kernelReady_st .................................................................................... 186
blockDim ..................................................................................................................................... 187
cntor ........................................................................................................................................... 187
dev ..............................................................................................................................................187
function ..................................................................................................................................... 187
functionEntry ............................................................................................................................. 187
gridDim ...................................................................................................................................... 187
gridld ..........................................................................................................................................187
module .......................................................................................................................................187
parentGridld .............................................................................................................................. 187
tid ............................................................................................................................................... 188
type ............................................................................................................................................ 188
CUDBGEventCallbackData ........................................................................................................... 188
tid ............................................................................................................................................... 188
timeout ....................................................................................................................................... 188
CUDBGEventCallbackData40 ....................................................................................................... 188
tid ............................................................................................................................................... 188
CUDBGGridInfo ............................................................................................................................. 188
blockDim ....................................................................................................................................189
cntor ........................................................................................................................................... 189
Chapter 5. Data Fields..................................................................................................................190
Chapter 6. Deprecated List.........................................................................................................200
Chapter 1. Release Notes

1.1. 11.8 Release

**New Unified Debugger backend**

A new debugger backend named the Unified Debugger (UD) has been introduced on Linux platforms with this release. UD is supported across multiple platforms including both Windows and Linux. The UD should mostly be transparent to existing clients of the API. The previous debugger backend, known as the classic debugger backend, can still be used by setting the environment variable `CUDBG_USE_LEGACY_DEBUGGER` to 1. UD is not supported on Maxwell GPUs. The clients of the API shall switch to the classic backend if Maxwell support is required.

**Device side cudaDeviceSynchronize() undefined behavior**

The clients of the API shall prevent the use of SingleStepWarp in the deprecated `cudaDeviceSynchronize()` function. Instead, revert to stepping over the call with a BP set and resume.

**CUDBG_EVENT_KERNEL_READY events are no longer delivered for GPU-launched grids**

`CUDBG_EVENT_KERNEL_READY` events for GPU-launched grids that were delivered over the ASYNC event pipe will no longer be sent. GPU-launched here refers to codes making use of CUDA Dynamic Parallelism. The existing implementation for this use case was imprecise. The callback did not report all GPU-launched grids before execution has begun, only those found on the device currently executing that were not previously reported during their launch. This functionality may be reintroduced in a future release. If this functionality is strictly required, the classic debugger backend can be used.

**getLoadedFunctionInfo**

Added a new `getLoadedFunctionInfo` call to obtain the section number and address of loaded functions for a given module.

1.2. 7.0 Release

**Stability improvements. No API additions or changes.**
1.3. 6.5 Release

Predicate registers
The per-thread predicate registers can be accessed and modified via the readPredicates() and writePredicates() calls. Each of these calls expects a buffer of sufficient size to cover all predicates for the current GPU architecture. The number of current predicate registers can be read back via the getNumPredicates() API call.

Condition code register
The per-thread condition code register can be accessed and modified via the readCCRegister() and writeCCRegister() calls. The condition code register is a unsigned 32-bit register, whose format may vary by GPU architecture.

Device Name
The getDeviceName() API returns a string containing the publically exposed product name of the GPU.

API Error Reporting Improvement
The symbol CUDBG_REPORT_DRIVER_API_ERROR_FLAGS points to an unsigned 32-bit integer in the application’s process space that controls API error reporting. The values that can be written into this flag are specified in the CUDBGReportDriverApiErrorFlags enum. In 6.5, setting the bit corresponding to CUDBG_REPORT_DRIVER_API_ERROR_FLAGS_SUPPRESS_NOT_READY in the variable CUDBG_REPORT_DRIVER_API_ERROR_FLAGS is supported. This will prevent CUDA API calls that return the runtime API error code cudaErrorNotReady or the driver API error code cuErrorNotReady from executing the CUDA API error reporting function.
Chapter 2. 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.

2.1. Debugger API

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

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

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

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

Only when stopped, can we query the machine’s state. Warp state includes which function is it runnning, which block, which lanes are valid, etc.
As of CUDA 6.0, state collection functions in the debug API will return
CUDA_ERROR_RUNNING_DEVICE if called without first calling the suspendDevice entry
point to ensure the device is stopped.

Clients of the debug API should suspend all devices before servicing a CUDBGEvent. A valid
CUDBGEvent is only guaranteed to be returned after the notification callback set using
CUDBGAPI_st::setNotifyNewEventCallback() is executed. Any debug API entry point will return
CUDA_ERROR_RECURSIVE_API_CALL when the call is made from within the notification
callback set using CUDBGAPI_st::setNotifyNewEventCallback().

2.2. ELF and DWARF

CUDA applications are compiled in ELF binary format.

Starting with CUDA 6.0, DWARF device information is obtained through an API call of
CUDBGAPI_st::getElfImageByHandle using the handle exposed from CUDBGEvent
of type CUDBG_EVENT_ELF_IMAGE_LOADED. This means that the information is
not available until runtime, after the CUDA driver has loaded. The DWARF device
information lifetime is valid until it is unloaded, which presents a CUDBGEvent of type
CUDBG_EVENT_ELF_IMAGE_UNLOADED.

In CUDA 5.5 and earlier, the DwarF device information was returned as part of the
CUDBGEvent of type CUDBG_EVENT_ELF_IMAGE_LOADED. The pointers presented in
CUDBGEvent55 were read-only pointers to memory managed by the debug API. The memory
pointed to was implicitly scoped to the lifetime of the loading CUDA context. Accessing the
returned pointers after the context was destroyed resulted in undefined behavior.

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

- **CUDBGAPI\_st::readVirtualPC()**

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

```
<2><321>: Abbrev Number: 18 (DW\_TAG\_formal\_parameter)
   DW\_AT\_decl\_file   : 27
   DW\_AT\_decl\_line   : 5
   DW\_AT\_name        : res
   DW\_AT\_type        : <2c6>
   DW\_AT\_location    : 9 byte block: 3 18 0 0 0 0 0 0 0       (DW\_OP\_addr: 18)
   DW\_AT\_address\_class: 7
```

The above shows that variable ‘res’ has an address class of 7 [ptxParamStorage]. Its location information shows it is located at address 18 within the parameter memory segment.

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

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

```
<3><359>: Abbrev Number: 20 (DW\_TAG\_variable)
   DW\_AT\_decl\_file   : 27
   DW\_AT\_decl\_line   : 7
   DW\_AT\_name        : c
   DW\_AT\_type        : <1aa>
   DW\_AT\_location    : 7 byte block: 90 b9 e2 90 b3 d6 4      (DW\_OP\_regx: 160631632185)
   DW\_AT\_address\_class: 2
```

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

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

- **CUDBGAPI\_st::readPC()**
2.3. 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”.

2.4. 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).

2.5. Attaching and Detaching

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

1. Attach to the CPU process corresponding to the CUDA application. The CPU part of the application will be frozen at this point.
2. Check to see if the CUDBG_IPC_FLAG_NAME variable is accessible from the memory space of the application. If not, it implies that the application has not loaded the CUDA driver, and the attaching to the application is complete.
3. Make a dynamic (inferior) function call to the function cudbgApInit() with an argument of "2", i.e., “cudbgApInit(2)”, e.g. by using ptrace(2) on Linux. This causes a helper process to be forked off from the application, which assists in attaching to the CUDA process.
4. Ensure that the initialization of the CUDA debug API is complete, or wait till API initialization is successful [i.e. call the “initialize()” API method until it succeeds].

5. Make the “initializeAttachStub()” API call to initialize the helper process that was forked off from the application earlier.

6. Read the value of the CUDBG_RESUME_FOR_ATTACH_DETACH variable from the memory space of the application:
   - If the value is non-zero, resume the CUDA application so that more data can be collected about the application and sent to the debugger. When the application is resumed, the debug client can expect to receive various CUDA events from the CUDA application. Once all state has been collected, the debug client will receive the event CUDBG_EVENT_ATTACH_COMPLETE.
   - If the value is zero, there is no more attach data to collect. Set the CUDBG_IPC_FLAG_NAME variable to 1 in the application’s process space, which enables further events from the CUDA application.

7. At this point, attaching to the CUDA application is complete and all GPUs belonging to the CUDA application will be suspended.

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

1. Check to see if the CUDBG_IPC_FLAG_NAME variable is accessible from the memory space of the application, and that the CUDA debug API is initialized. If either of these conditions is not met, treat the application as CPU-only and detach from the application.

2. Next, make the “clearAttachState” API call to prepare the CUDA debug API for detach.

3. Make a dynamic (inferior) function call to the function cudbgApiDetach() in the memory space of the application, e.g. by using ptrace(2) on Linux. This causes CUDA driver to setup state for detach.

4. Read the value of the CUDBG_RESUME_FOR_ATTACH_DETACH variable from the memory space of the application. If the value is non-zero, make the “requestCleanupOnDetach” API call.

5. Set the CUDBG_DEBUGGER_INITIALIZED variable to 0 in the memory space of the application. This makes sure the debugger is reinitialized from scratch if the debug client re-attaches to the application in the future.

6. If the value of the CUDBG_RESUME_FOR_ATTACH_DETACH variable was found to be non-zero in step 4, delete all breakpoints and resume the CUDA application. This allows the CUDA driver to perform cleanups before the debug client detaches from it. Once the cleanup is complete, the debug client will receive the event CUDBG_EVENT_DETACH_COMPLETE.

7. Set the CUDBG_IPC_FLAG_NAME variable to zero in the memory space of the application. This prevents any more callbacks from the CUDA application to the debugger.
8. The client must then finalize the CUDA debug API.

9. Finally, detach from the CPU part of the CUDA application. At this point all GPUs belonging to the CUDA application will be resumed.
Chapter 3. Modules

Here is a list of all modules:

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

3.1. General

`enum CUDBGResult`

Result values of all the API routines.

**Values**

- `CUDBG_SUCCESS = 0x0000`
  - The API call executed successfully.
- `CUDBG_ERROR_UNKNOWN = 0x0001`
  - Error type not listed below.
- `CUDBG_ERROR_BUFFER_TOO_SMALL = 0x0002`
  - Cannot copy all the queried data into the buffer argument.
- `CUDBG_ERROR_UNKNOWN_FUNCTION = 0x0003`
  - Function cannot be found in the CUDA kernel.
CUDBG_ERROR_INVALID_ARGS = 0x0004
  Wrong use of arguments [NULL pointer, illegal value, ...].

CUDBG_ERROR_UNINITIALIZED = 0x0005
  Debugger API has not yet been properly initialized.

CUDBG_ERROR_INVALID_COORDINATES = 0x0006
  Invalid block or thread coordinates were provided.

CUDBG_ERROR_INVALID_MEMORY_SEGMENT = 0x0007
  Invalid memory segment requested.

CUDBG_ERROR_INVALID_MEMORY_ACCESS = 0x0008
  Requested address (+size) is not within proper segment boundaries.

CUDBG_ERROR_MEMORY_MAPPING_FAILED = 0x0009
  Memory is not mapped and cannot be mapped.

CUDBG_ERROR_INTERNAL = 0x000a
  A debugger internal error occurred.

CUDBG_ERRORINVALID_DEVICE = 0x000b
  Specified device cannot be found.

CUDBG_ERROR_INVALID_SM = 0x000c
  Specified sm cannot be found.

CUDBG_ERROR_INVALID_WARP = 0x000d
  Specified warp cannot be found.

CUDBG_ERROR_INVALID_LANE = 0x000e
  Specified lane cannot be found.

CUDBG_ERROR_SUSPENDED_DEVICE = 0x000f
  The requested operation is not allowed when the device is suspended.

CUDBG_ERROR_RUNNING_DEVICE = 0x0010
  Device is running and not suspended.

CUDBG_ERROR_RESERVED_0 = 0x0011

CUDBG_ERROR_INVALID_ADDRESS = 0x0012
  Address is out-of-range.

CUDBG_ERROR_INCOMPATIBLE_API = 0x0013
  The requested API is not available.

CUDBG_ERROR_INITIALIZATION_FAILURE = 0x0014
  The API could not be initialized.

CUDBG_ERROR_INVALID_GRID = 0x0015
  The specified grid is not valid.

CUDBG_ERROR_NO_EVENT_AVAILABLE = 0x0016
  The event queue is empty and there is no event left to be processed.

CUDBG_ERROR_SOME_DEVICES_WATCHDOGGED = 0x0017
  Some devices were excluded because they have a watchdog associated with them.

CUDBG_ERROR_ALL_DEVICES_WATCHDOGGED = 0x0018
  All devices were exclude because they have a watchdog associated with them.

CUDBG_ERROR_INVALID_ATTRIBUTE = 0x0019
  Specified attribute does not exist or is incorrect.
CUDBG_ERROR_ZERO_CALL_DEPTH = 0x001a
No function calls have been made on the device.

CUDBG_ERROR_INVALID_CALL_LEVEL = 0x001b
Specified call level is invalid.

CUDBG_ERROR_COMMUNICATION_FAILURE = 0x001c
Communication error between the debugger and the application.

CUDBG_ERROR_INVALID_CONTEXT = 0x001d
Specified context cannot be found.

CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM = 0x001e
Requested address was not originally allocated from device memory [most likely visible in system memory].

CUDBG_ERROR_MEMORY_UNMAPPING_FAILED = 0x001f
Requested address is not mapped and cannot be unmapped.

CUDBG_ERROR_INCOMPATIBLE_DISPLAY_DRIVER = 0x0020
The display driver is incompatible with the API.

CUDBG_ERROR_INVALID_MODULE = 0x0021
The specified module is not valid.

CUDBG_ERROR_LANE_NOT_IN_SYSCALL = 0x0022
The specified lane is not inside a device syscall.

CUDBG_ERROR_MEMCHECK_NOT_ENABLED = 0x0023
Memcheck has not been enabled.

CUDBG_ERROR_INVALID_ENVVAR_ARGS = 0x0024
Some environment variable’s value is invalid.

CUDBG_ERROR_OS_RESOURCES = 0x0025
Error while allocating resources from the OS.

CUDBG_ERROR_FORK_FAILED = 0x0026
Error while forking the debugger process.

CUDBG_ERROR_NO_DEVICE_AVAILABLE = 0x0027
No CUDA capable device was found.

CUDBG_ERROR_ATTACH_NOT_POSSIBLE = 0x0028
Attaching to the CUDA program is not possible.

CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE = 0x0029

CUDBG_ERROR_INVALID_WARP_MASK = 0x002a

CUDBG_ERROR_AMBIGUOUS_MEMORY_ADDRESS = 0x002b
Specified device pointer cannot be resolved to a GPU unambiguously because it is valid on more than one GPU.

CUDBG_ERROR_RECURSIVE_API_CALL = 0x002c

CUDBG_ERROR_MISSING_DATA = 0x002d

CUDBG_ERROR_NOT_SUPPORTED = 0x002e

3.2. Initialization
CUDBGResult (*CUDBGAPI_st::finalize) ()
Finalize the API and free all memory.

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_COMMUNICATION_FAILURE, CUDBG_ERROR_UNKNOWN
Since CUDA 3.0.

See also:
initialize

CUDBGResult (*CUDBGAPI_st::initialize) ()
Initialize the API.

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNKNOWN
Since CUDA 3.0.

See also:
finalize

3.3. Device Execution Control

CUDBGResult (*CUDBGAPI_st::resumeDevice) (uint32_t dev)
Resume a suspended CUDA device.

Parameters
dev
- device index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.
See also:
suspendDevice
singleStepWarp

CUDAResult (*CUDBGAPI_st::resumeWarpsUntilPC) (uint32_t devId, uint32_t sm, uint64_t warpMask, uint64_t virtPC)

Inserts a temporary breakpoint at the specified virtual PC, and resumes all warps in the specified bitmask on a given SM. As compared to CUDBGAPI_st::resumeDevice, CUDBGAPI_st::resumeWarpsUntilPC provides finer-grain control by resuming a selected set of warps on the same SM. The main intended usage is to accelerate the single-stepping process when the target PC is known in advance. Instead of single-stepping each warp individually until the target PC is hit, the client can issue this API. When this API is used, errors within CUDA kernels will no longer be reported precisely. In the situation where resuming warps is not possible, this API will return CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE. The client should then fall back to using CUDBGAPI_st::singleStepWarp or CUDBGAPI_st::resumeDevice.

Parameters

devId
- device index

sm
- the SM index

warpMask
- the bitmask of warps to resume (1 = resume, 0 = do not resume)

virtPC
- the virtual PC where the temporary breakpoint will be inserted

Returns

CUDBG_SUCCESS CUDBG_ERROR_INVALID_ARGS CUDBG_ERROR_INVALID_DEVICE
CUDBG_ERROR_INVALID_SM CUDBG_ERROR_INVALID_WARP_MASK
CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE CUDBG_ERROR_UNINITIALIZED

Since CUDA 6.0.

See also:
resumeDevice
CUDBGResult (*CUDBGAPI_st::singleStepWarp) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t nsteps, uint64_t *warpMask)

Single step an individual warp nsteps times on a suspended CUDA device. Only the last instruction in a range should be a control flow instruction.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

nsteps
- number of single steps

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

Since CUDA 7.5.

See also:
resumeDevice
suspendDevice

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

Single step an individual warp on a suspended CUDA device.

Parameters

dev
- device index

sm
- SM index
wp
- warp index

**Returns**
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM,
CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_RUNNING_DEVICE,
CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_UNKNOWN,
CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE

Since CUDA 3.0.
[Deprecated](#) in CUDA 4.1.

**See also:**
resumeDevice
suspendDevice
singleStepWarp

```
CUDBGResult (*CUDBGAPI_st::singleStepWarp41)(
    uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *warpMask)
```
Single step an individual warp on a suspended CUDA device.

**Parameters**

- **dev**
  - device index

- **sm**
  - SM index

- **wp**
  - warp index

- **warpMask**
  - the warps that have been single-stepped

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

Since CUDA 4.1.
[Deprecated](#) in CUDA 7.5.
See also:
resumeDevice
suspendDevice

CUDBGResult (*CUDBGAPI_st::suspendDevice) (uint32_t dev)
Suspends a running CUDA device.

Parameters
dev
- device index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
resumeDevice
singleStepWarp

3.4. Breakpoints

CUDBGResult (*CUDBGAPI_st::getAdjustedCodeAddress) (uint32_t devId, uint64_t address, uint64_t *adjustedAddress, CUDBGAdjAddrAction adjAction)
The client must call this function before inserting a breakpoint, or when the previous or next code address is needed. Returns the adjusted code address for a given code address for a given device.

Parameters
devId
- the device index
address

adjustedAddress
- adjusted address

adjAction
- whether the adjusted next, previous or current address is needed

Returns

CUDA_SUCCESS, CUDA_ERROR_UNINITIALIZED, CUDA_ERROR_INVALID_ADDRESS, CUDA_ERROR_INVALID_DEVICE

Since CUDA 5.5.

See also:

unsetBreakpoint

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

Sets a breakpoint at the given instruction address for the given device. Before setting a breakpoint, CUDBGAPI_st::getAdjustedCodeAddress should be called to get the adjusted breakpoint address.

Parameters

dev
- the device index

addr
- instruction address

Returns

CUDA_SUCCESS, CUDA_ERROR_UNINITIALIZED, CUDA_ERROR_INVALID_ADDRESS, CUDA_ERROR_INVALID_DEVICE

Since CUDA 3.2.

See also:

unsetBreakpoint
CUDBGResult (*CUDBGAPI_st::setBreakpoint31) (uint64_t addr)
Sets a breakpoint at the given instruction address.

Parameters
addr
- instruction address

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS
Since CUDA 3.0.
Deprecated in CUDA 3.2.

See also:
unsetBreakpoint31

CUDBGResult (*CUDBGAPI_st::unsetBreakpoint) (uint32_t dev, uint64_t addr)
Unsets a breakpoint at the given instruction address for the given device.

Parameters
dev
- the device index
addr
- instruction address

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS,
CUDBG_ERROR_INVALID_DEVICE
Since CUDA 3.2.

See also:
setBreakpoint
CUDBGResult (*CUDBGAPI_st::unsetBreakpoint31) (uint64_t addr)
Unsets a breakpoint at the given instruction address.

Parameters

addr
- instruction address

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.
Deprecated in CUDA 3.2.

See also:
setBreakpoint31

3.5. Device State Inspection

CUDBGResult (*CUDBGAPI_st::getLoadedFunctionInfo) (uint32_t devId, uint64_t handle, CUDBGLoadedFunctionInfo *info, uint32_t numEntries)
Get the section number and address of loaded functions for a given module.

Parameters

devId
handle
- ELF/cubin image handle
info
numEntries
- number of function load entries to read

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_NOT_SUPPORTED
Since CUDA 11.8
CUDBGResult (*CUDBGAPI_st::getManagedMemoryRegionInfo) (uint64_t startAddress, CUDBGMemoryInfo *memoryInfo, uint32_t memoryInfo_size, uint32_t *numEntries)

Returns a sorted list of managed memory regions. The sorted list of memory regions starts from a region containing the specified starting address. If the starting address is set to 0, a sorted list of managed memory regions is returned which starts from the managed memory region with the lowest start address.

Parameters

**startAddress**
- The address that the first region in the list must contain. If the starting address is set to 0, the list of managed memory regions returned starts from the managed memory region with the lowest start address.

**memoryInfo**
- Client-allocated array of memory region records of type CUDBGMemoryInfo.

**memoryInfo_size**
- Number of records of type CUDBGMemoryInfo that memoryInfo can hold.

**numEntries**
- Pointer to a client-allocated variable holding the number of valid entries returned in memoryInfo. Valid entries are contiguous and start from memoryInfo[0].

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INTERNAL

Since CUDA 6.0.
CUDBGResult (*CUDBGAPI_st::memcheckReadErrorAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *address, ptxStorageKind *storage)
Get the address that memcheck detected an error on.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
address
  - returned address detected by memcheck
storage
  - returned address class of address

Returns
CUDBG_ERROR_NOT_SUPPORTED,
Since CUDA 5.0.

CUDBGResult (*CUDBGAPI_st::readActiveLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *activeLanesMask)
Reads the bitmask of active lanes on a valid warp.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
activeLanesMask
- the returned bitmask of active lanes

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE,
CUDA_ERROR_INVALID_SM, CUDA_ERROR_INVALID_WARP,
CUDA_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
readGridId
readBlockIdx
readThreadId
readBrokenWarps
readValidWarps
readValidLanes

CUDA_dbgResult (*CUDA_dbgAPI_st::readBlockIdx)
(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockIdx)
Reads the CUDA block index running on a valid warp.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
blockIdx
- the returned CUDA block index

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE,
CUDA_ERROR_INVALID_SM, CUDA_ERROR_INVALID_WARP,
CUDA_ERROR_UNINITIALIZED
Since CUDA 4.0.
See also:
readGridId
readThreadIdx
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

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

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

**Parameters**

dev
  - device index

sm
  - SM index

wp
  - warp index

blockIdx
  - the returned CUDA block index

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

[Deprecated](#) in CUDA 4.0.

See also:
readGridId
readThreadIdx
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

CUDBGResult (*CUDBGAPI_st::readBrokenWarps)(uint32_t dev, uint32_t sm, uint64_t *brokenWarpsMask)
Reads the bitmask of warps that are at a breakpoint on a given SM.

Parameters

*dev
  - device index
*sm
  - SM index
*brokenWarpsMask
  - the returned bitmask of broken warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:

readGridId
readBlockId
readThreadId
readValidWarps
readValidLanes
readActiveLanes
CUDBGResult (*CUDBGAPI_st::readCallDepth)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)
Reads the call depth [number of calls] for a given lane.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
depth
- the returned call depth

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID.Args, CUDBG_ERROR_INVALID.DEVICE,
CUDBG_ERROR_INVALID.SM, CUDBG_ERROR_INVALID.WARP,
CUDBG_ERROR_INVALID.LANE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 4.0.

See also:
readReturnAddress
readVirtualReturnAddress

CUDBGResult (*CUDBGAPI_st::readCallDepth32)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)
Reads the call depth [number of calls] for a given warp.

Parameters
dev
- device index
sm
- SM index
wp
- warp index

depth
- the returned call depth

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.1.
Deprecated in CUDA 4.0.

See also:
readReturnAddress32
readVirtualReturnAddress32

CUDBGResult (*CUDBGAPI_st::readCCRegister)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *val)
Reads the hardware CC register.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane 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

Since CUDA 6.5.
See also:
readCodeMemory
readConstMemory
readGenericMemory
readGlobalMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
readPredicates

CUDBGResult (*CUDBGAPI_st::readClusterIdx)(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *clusterIdx)
Reads the CUDA cluster index running on a valid warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

clusterIdx
- the returned CUDA cluster index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 12.0.

See also:
readGridId
**readThreadIdx**
**readBlockIdx**
**readBrokenWarps**
**readValidWarps**
**readValidLanes**
**readActiveLanes**

**CUDBGResult (*CUDBGAPI_st::readCodeMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)**
Reads content at address in the code memory segment.

**Parameters**
- **dev**
  - device index
- **addr**
  - memory address
- **buf**
  - buffer
- **sz**
  - size of the buffer

**Returns**
- CUDBG_SUCCESS
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_DEVICE
- CUDBG_ERROR_UNINITIALIZED
- CUDBG_ERROR_MEMORY_MAPPING_FAILED

Since CUDA 3.0.

**See also:**
- **readConstMemory**
- **readGenericMemory**
- **readParamMemory**
- **readSharedMemory**
- **readTextureMemory**
- **readLocalMemory**
- **readRegister**
- **readPC**
CUDBGResult (*CUDBGAPI_st::readConstMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the constant memory segment.

Parameters

dev
- device index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, 
CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Since CUDA 3.0.

See also:
readCodeMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
CUDBGResult (*CUDBGAPI_st::readErrorPC)(uint32_t devId, uint32_t sm, uint32_t wp, uint64_t *errorPC, bool *errorPCValid)

Get the hardware reported error PC if it exists.

Parameters

- devId - the device index
- sm - the SM index
- wp - warp index
- errorPC - PC of the exception
- errorPCValid - boolean to indicate that the returned error PC is valid

Returns

- CUDBG_SUCCESS
- CUDBG_ERROR_UNINITIALIZED
- CUDBG_ERROR_INVALID_DEVICE
- CUDBG_ERROR_INVALID_SM
- CUDBG_ERROR_INVALID_WARP
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_UNKNOWN_FUNCTION

Since CUDA 6.0

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

Reads content at an address in the generic address space. This function determines if the given address falls into the local, shared, or global memory window. It then accesses memory taking into account the hardware co-ordinates provided as inputs.

Parameters

- dev - device index
- sm - SM index
- wp - warp index
- ln - lane index
addr
- memory address

buf
- buffer

sz

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

Since CUDA 6.0.

See also:
readCodeMemory
readConstMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC

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

Reads content at an address in the global address space. If the address is valid on more than one device and one of those devices does not support UVA, an error is returned.

Parameters
addr
- memory address

buf
- buffer
**Returns**

CUDA_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_INVALID_MEMORY_ACCESS, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM, CUDBG_ERROR_AMBIGUOUS_MEMORY_ADDRESS_

Since CUDA 6.0.

**See also:**

readCodeMemory
readConstMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC

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

Reads content at address in the global memory segment.

**Parameters**

**dev**
- device index

**addr**
- memory address

**buf**
- buffer

**sz**
- size of the buffer

**Returns**

CUDA_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED
Since CUDA 3.0. 
**Deprecated** in CUDA 3.2.

See also:
- `readCodeMemory`
- `readConstMemory`
- `readParamMemory`
- `readSharedMemory`
- `readTextureMemory`
- `readLocalMemory`
- `readRegister`
- `readPC`

```c
CUDBGResult (*CUDBGAPI_st::readGlobalMemory55) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)
```

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

**Parameters**
- `dev` - device index
- `sm` - SM index
- `wp` - warp index
- `ln` - lane index
- `addr` - memory address
- `buf` - buffer
- `sz` - size of the buffer

**Returns**
- `CUDBG_SUCCESS`
- `CUDBG_ERROR_INVALID_ARGS`
- `CUDBG_ERROR_INVALID_DEVICE`
- `CUDBG_ERROR_INVALID_LANE`
- `CUDBG_ERROR_INVALID_SM`
CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED,
CUDBG_ERROR_MEMORY_MAPPING_FAILED,
CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Since CUDA 3.2.
Deprecated in CUDA 6.0.

See also:
readCodeMemory
readConstMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC

CUDBGResult (*CUDBGAPI_st::readGridId) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *gridId64)
Reads the 64-bit CUDA grid index running on a valid warp.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
gridId64

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 5.5.

See also:
CUDBGResult (*CUDBGAPI_st::readGridId50) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)
Reads the CUDA grid index running on a valid warp.

Parameters
- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **gridId**
  - the returned CUDA grid index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.
**Deprecated** in CUDA 5.5.

See also:
- readBlockIdx
- readThreadIdx
- readBrokenWarps
- readValidWarps
- readValidLanes
- readActiveLanes
readActiveLanes

CUDBGResult (*CUDBGAPI_st::readLaneException) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, CUDBGException_t *exception)
Reads the exception type for a given lane.

Parameters

dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
exception
- the returned exception type

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

Since CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::readLaneStatus) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, bool *error)
Reads the status of the given lane. For specific error values, use readLaneException.

Parameters

dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index

error
- true if there is an error

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

Since CUDA 3.0.

CUDBGResult (*CUDBGAPI_st::readLocalMemory) 
(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, 
uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the local memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

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

Since CUDA 3.0.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readRegister
readPC

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

Reads content at address in the param memory segment.

Parameters

dev
   - device index

sm
   - SM index

wp
   - warp index

addr
   - memory address

buf
   - buffer

sz
   - size of the buffer

Returns

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

Since CUDA 3.0.

See also:

readCodeMemory
readConstMemory
CUDBGResult (*CUDBGAPI_st::readPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)
Reads the PC on the given active lane.

Parameters

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**ln**
- lane index

**pc**
- the returned PC

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
**readLocalMemory**

**readRegister**

**readVirtualPC**


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

Reads content at pinned address in system memory.

**Parameters**

**addr**
- system memory address

**buf**
- buffer

**sz**
- size of the buffer

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, 
CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.2.

**See also:**

readCodeMemory

readConstMemory

readGenericMemory

readParamMemory

readSharedMemory

readTextureMemory

readLocalMemory

readRegister

readPC
CUDBGResult (*CUDBGAPI_st::readPredicates)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t predicates_size, uint32_t *predicates)
Reads content of hardware predicate registers.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

predicates_size
- number of predicate registers to read

predicates
- 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

Since CUDA 6.5.

See also:

readCodeMemory
readConstMemory
readGenericMemory
readGlobalMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
CUDBGResult (*CUDBGAPI_st::readRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t *val)

Reads content of a hardware register.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
regno
- register index
val
- buffer

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

Since CUDA 3.0.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readPC
CUDBGResult (*CUDBGAPI_st::readRegisterRange) (uint32_t devId, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t index, uint32_t registers_size, uint32_t *registers)

Reads content of a hardware range of hardware registers.

Parameters

- **devId**
- **sm**
  - SM index
- **wp**
  - warp index
- **ln**
  - lane index
- **index**
  - index of the first register to read
- **registers_size**
  - number of registers to read
- **registers**
  - 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

Since CUDA 6.0.

See also:

- readCodeMemory
- readConstMemory
- readGenericMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readLocalMemory
- readPC
readRegister

CUDBGResult (*CUDBGAPI_st::readReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)
Reads the physical return address for a call level.

Parameters

- **dev**  
  - device index
- **sm**  
  - SM index
- **wp**  
  - warp index
- **ln**  
  - lane index
- **level**  
  - the specified call level
- **ra**  
  - the returned return address for level

Returns

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

Since CUDA 4.0.

See also:

- readCallDepth
- readVirtualReturnAddress
CUDBGResult

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

Reads the physical return address for a call level.

Parameters

- `dev` - device index
- `sm` - SM index
- `wp` - warp index
- `level` - the specified call level
- `ra` - the returned return address for level

Returns

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

Since CUDA 3.1.

Deprecated in CUDA 4.0.

See also:

readCallDepth32
readVirtualReturnAddress32
CUDBGResult (*CUDBGAPI_st::readSharedMemory)(uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the shared memory segment.

Parameters

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**addr**
- memory address

**buf**
- buffer

**sz**
- size of the buffer

Returns

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

Since CUDA 3.0.

See also:
- readCodeMemory
- readConstMemory
- readGenericMemory
- readParamMemory
- readLocalMemory
- readTextureMemory
- readRegister
- readPC
CUDBGResult (*CUDBGAPI_st::readSyscallCallDepth)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)
Reads the call depth of syscalls for a given lane.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
depth
- the returned call depth

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

Since CUDA 4.1.

See also:
readReturnAddress
readVirtualReturnAddress

CUDBGResult (*CUDBGAPI_st::readTextureMemory)(uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t id,
uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)
This method is no longer supported since CUDA 12.0.

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
ds
  - size of the buffer

Returns
CUDBG_ERROR_NOT_SUPPORTED,

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readLocalMemory
readRegister
readPC

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

This method is no longer supported since CUDA 12.0.

Parameters
devId
  - device index
CUDA Debugger API

TRM-06710-001 _vRelease Version | 49

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_ERROR_NOT_SUPPORTED

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readLocalMemory
readRegister
readPC

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

Reads the CUDA thread index running on valid lane.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index

ln
  - lane index

threadIdx
  - the returned CUDA thread index

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

Since CUDA 3.0.

**See also:**
readGridId
readBlockIdx
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

**CUDBGResult**
(*CUDBGAPI_st::readUniformPredicates)*

**Parameters**

*dev*
  - device index

*sm*
  - SM index

*wp*
  - warp index

*predicates_size*
  - number of predicate registers to read

*predicates*
  - buffer
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED
Since CUDA 10.0.

See also:
readPredicates

CUDBGResult
(*CUDBGAPI_st::readUniformRegisterRange)
(uint32_t devId, uint32_t sm, uint32_t wp, uint32_t regno, uint32_t registers_size, uint32_t *registers)
Reads a range of uniform registers.

Parameters
devId
sm
- SM index
wp
- warp index
regno
- starting index into uniform register file
registers_size
- number of bytes to read
registers
- pointer to buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED
Since CUDA 10.0.

See also:
readRegister
CUDBGResult (*CUDBGAPI_st::readValidLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *validLanesMask)
Reads the bitmask of valid lanes on a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

validLanesMask
- the returned bitmask of valid lanes

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
readGridId
readBlockIdx
readThreadIdx
readBrokenWarps
readValidWarps
readActiveLanes

CUDBGResult (*CUDBGAPI_st::readValidWarps) (uint32_t dev, uint32_t sm, uint64_t *validWarpsMask)
Reads the bitmask of valid warps on a given SM.

Parameters

dev
- device index
sm
  - SM index
validWarpsMask
  - the returned bitmask of valid warps

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
readGridId
readBlockIdx
readThreadIdx
readBrokenWarps
readValidLanes
readActiveLanes

CUDBGResult (*CUDBGAPI_st::readVirtualPC)
(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln,
uint64_t *pc)
Reads the virtual PC on the given active lane.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
pc
  - the returned PC

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

Since CUDA 3.0.

See also:
readPC

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

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

level
- the specified call level

ra
- the returned virtual return address for level

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALIDDEVICE, 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

Since CUDA 4.0.

See also:
readCallDepth
**readReturnAddress**

**CUDBGResult readReturnAddress32:**

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

Reads the virtual return address for a call level.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **level**
  - the specified call level
- **ra**
  - the returned virtual return address for level

**Returns**

- CUDA_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

Since CUDA 3.1.

**Deprecated** in CUDA 4.0.

**See also:**

- readCallDepth32
- readReturnAddress32
CUDBGResult (*CUDBGAPI_st::readWarpState) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGWarpState *state)
Get state of a given warp.

Parameters

dev
- device index
sm
- SM index
wp
- warp index
state
- pointer to structure that contains warp status

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED,
Since CUDA 12.0.

CUDBGResult (*CUDBGAPI_st::readWarpState60) (uint32_t devId, uint32_t sm, uint32_t wp, CUDBGWarpState60 *state)
Get state of a given warp.

Parameters

devId
sm
- SM index
wp
- warp index
state
- pointer to structure that contains warp status
Returns
CUDA_DBG_SUCCESS, CUDA_DBG_ERROR_INVALID_ARGS, CUDA_DBG_ERROR_INVALID_DEVICE,
CUDA_DBG_ERROR_INVALID_SM, CUDA_DBG_ERROR_INVALID_WARP,
CUDA_DBG_ERROR_UNINITIALIZED,

Since CUDA 6.0.

CUDA_DBGResult (*CUDA_DBGAPI_st::writePinnedMemory)(uint64_t addr, const void *buf, uint32_t sz)
Writes content to pinned address in system memory.

Parameters
addr
- system memory address
buf
- buffer
sz
- size of the buffer

Returns
CUDA_DBG_SUCCESS, CUDA_DBG_ERROR_INVALID_ARGS,
CUDA_DBG_ERROR_MEMORY_MAPPING_FAILED, CUDA_DBG_ERROR_UNINITIALIZED

Since CUDA 3.2.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readLocalMemory
readRegister
readPC
CUDBGResult (*CUDBGAPI_st::writePredicates)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t predicates_size, const uint32_t *predicates)

Writes content to hardware predicate registers.

Parameters

- **dev** - device index
- **sm** - SM index
- **wp** - warp index
- **ln** - lane index
- **predicates_size** - number of predicate registers to write
- **predicates** - 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

Since CUDA 6.5.

See also:

- writeConstMemory
- writeGenericMemory
- writeGlobalMemory
- writeParamMemory
- writeSharedMemory
- writeTextureMemory
- writeLocalMemory
- writeRegister
CUDBGResult (*CUDBGAPI_st::writeUniformPredicates) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t predicates_size, const uint32_t *predicates)

Writes to uniform predicate registers.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **predicates_size**
  - number of predicate registers to write
- **predicates**
  - buffer

**Returns**

- CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_INVALID_SM, CUDA_ERROR_INVALID_WARP, CUDA_ERROR_UNINITIALIZED

Since CUDA 10.0.

**See also:**

- readUniformPredicate
- writeRegister

### 3.6. Device State Alteration
CUDBGResult (*CUDBGAPI_st::writeCCRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t val)
Writes the hardware CC register.

Parameters

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**ln**
- lane index

**val**
- value to write to the CC register

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

Since CUDA 6.5.

See also:
writeConstMemory
writeGenericMemory
writeGlobalMemory
writeParamMemory
writeSharedMemory
writeTextureMemory
writeLocalMemory
writeRegister
writePredicates
CUDBGResult (*CUDBGAPI_st::writeGenericMemory)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)

Writes content to an address in the generic address space. This function determines if the given address falls into the local, shared, or global memory window. It then accesses memory taking into account the hardware co-ordinates provided as inputs.

Parameters

dev - device index
sm - SM index
wp - warp index
ln - lane index
addr - memory address
buf - buffer
sz

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

Since CUDA 6.0.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister
**CUDBGResult (*CUDBGAPI_st::writeGlobalMemory) (uint64_t addr, const void *buf, uint32_t sz)**

Writers content to an address in the global address space. If the address is valid on more than one device and one of those devices does not support UVA, an error is returned.

**Parameters**

- **addr**
  - memory address
- **buf**
  - buffer
- **sz**

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_INVALID_MEMORY_ACCESS, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM, CUDBG_ERROR_AMBIGUOUS_MEMORY_ADDRESS

Since CUDA 6.0.

**See also:**

- writeParamMemory
- writeSharedMemory
- writeLocalMemory
- writeRegister

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

Writes content to address in the global memory segment.

**Parameters**

- **dev**
  - device index
- **addr**
  - memory address
buf
  - buffer
sz
  - size of the buffer

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

Since CUDA 3.0.
Deprecated in CUDA 3.2.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister

CUDBGResult
(*CUDBGAPI_st::writeGlobalMemory55) (uint32_t dev,
uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr,
const void *buf, uint32_t sz)

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

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
addr
  - memory address
buf
  - buffer
sz
- size of the buffer

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE,
CUDA_ERROR_INVALID_LANE, CUDA_ERROR_INVALID_SM,
CUDA_ERROR_INVALID_WARP, CUDA_ERROR_UNINITIALIZED,
CUDA_ERROR_MEMORY_MAPPING_FAILED,
CUDA_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Since CUDA 3.2.
Deprecated in CUDA 6.0.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister

CUDBGResult (*CUDBGAPI_st::writeLocalMemory)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln,
uint64_t addr, const void *buf, uint32_t sz)
Writes content to address in the local memory segment.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
addr
- memory address
buf
- buffer
sz
- size of the buffer
Returns

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

Since CUDA 3.0.

See also:
writeGenericMemory
writeParamMemory
writeSharedMemory
writeRegister

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

Writes content to address in the param memory segment.

Parameters

dev
  - device index
sm
  - SM index
wp
  - warp index
addr
  - memory address
buf
  - buffer
sz
  - size of the buffer

Returns

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

Since CUDA 3.0.
CUDBGResult (*CUDBGAPI_st::writeRegister)
(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln,
uint32_t regno, uint32_t val)
Writes content to a hardware register.

Parameters

- **dev**
  - device index

- **sm**
  - SM index

- **wp**
  - warp index

- **ln**
  - lane index

- **regno**
  - register index

- **val**
  - buffer

Returns

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

Since CUDA 3.0.

See also:

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

Writes content to address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

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

Since CUDA 3.0.

See also:

writeGenericMemory
writeParamMemory
writeLocalMemory
writeRegister
CUDBGResult (*CUDBGAPI_st::writeUniformRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t regno, uint32_t val)
Writes content to a uniform register.

Parameters
\texttt{dev}
- device index
\texttt{sm}
- SM index
\texttt{wp}
- warp index
\texttt{regno}
- register index
\texttt{val}
- buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED
Since CUDA 10.0.

See also:
writeRegister
readUniformRegisterRange

3.7. Grid Properties

\textbf{struct} CUDBGGridInfo
Grid info.

\textbf{enum} CUDBGGridStatus
Grid status.
Values

CUDBG_GRID_STATUS_INVALID
An invalid grid ID was passed, or an error occurred during status lookup.

CUDBG_GRID_STATUS_PENDING
The grid was launched but is not running on the HW yet.

CUDBG_GRID_STATUS_ACTIVE
The grid is currently running on the HW.

CUDBG_GRID_STATUS_SLEEPING
The grid is on the device, doing a join.

CUDBG_GRID_STATUS_TERMINATED
The grid has finished executing.

CUDBG_GRID_STATUS_UNDETERMINED
The grid is either QUEUED or TERMINATED.

CUDBGResult (*CUDBGAPI_st::getBlockSize)
(uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)
Get the number of threads in the given block.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

blockDim
- the returned number of threads in the block

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:

getGridDim
CUDBGResult (*CUDBGAPI_st::getClusterDim)(uint32_t dev, uint64_t gridId64, CuDim3 *clusterDim)
Get the number of blocks in the given cluster.

Parameters

- **dev**
  - device index
- **gridId64**
  - grid ID
- **clusterDim**
  - the returned number of blocks in the cluster

Returns

- CUDBG_SUCCESS
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_GRID
- CUDBG_ERROR_UNINITIALIZED

Since CUDA 12.0.

See also:
getBlockDim
getGridDim

CUDBGResult (*CUDBGAPI_st::getElfImage)(uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint64_t *size)
Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **relocated**
  - set to true to specify the relocated ELF image, false otherwise
- **elfImage**
  - pointer to the ELF image
size
  - size of the ELF image (64 bits)

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, 
CUDBG_ERROR_UNINITIALIZED
Since CUDA 4.0.

CUDBGResult (*CUDBGAPI_st::getElfImage32)
  (uint32_t dev, uint32_t sm, uint32_t wp, bool 
   relocated, void **elfImage, uint32_t *size)
Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters
  dev
    - device index
  sm
    - SM index
  wp
    - warp index
  relocated
    - set to true to specify the relocated ELF image, false otherwise
  *elfImage
    - pointer to the ELF image
  size
    - size of the ELF image (32 bits)

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, 
CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.
Deprecated in CUDA 4.0.
**CUDBGResult (*CUDBGAPI_st::getGridAttribute) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttribute attr, uint64_t *value)**

Get the value of a grid attribute.

**Parameters**
- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **attr**
  - the attribute
- **value**
  - the returned value of the attribute

**Returns**
- CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.1.

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

Get several grid attribute values in a single API call.

**Parameters**
- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **pairs**
  - array of attribute/value pairs
- **numPairs**
  - the number of attribute/values pairs in the array
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, 
CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getGridDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *gridDim)
Get the number of blocks in the given grid.

Parameters
dev  
- device index
sm  
- SM index
wp  
- warp index
gridDim  
- the returned number of blocks in the grid

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, 
CUDBG_ERROR_UNINITIALIZED
Since CUDA 4.0.

See also:
getBlockDim

CUDBGResult (*CUDBGAPI_st::getGridDim32) 
(uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)
Get the number of blocks in the given grid.

Parameters
dev  
- device index
sm  
- SM index
**wp**
- warp index

**gridDim**
- the returned number of blocks in the grid

**Returns**
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.
[Deprecated](#) in CUDA 4.0.

**See also:**
getBlockDim

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

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

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

**Returns**
CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_SUCCESS

Since CUDA 12.0.
Get information about the specified grid. If the context of the grid has already been destroyed, the function will return CUDBG_ERROR_INVALID_GRID, although the grid id is correct.

Parameters

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

Returns

- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_GRID
- CUDBG_SUCCESS

Since CUDA 5.5.

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

Parameters

- `dev`
- `gridId64` - 64-bit grid ID
- `status` - enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns

- CUDBG_ERROR_INVALID_DEVICE
- CUDBG_ERROR_INVALID_GRID
- CUDBG_ERROR_INVALID_GRID

Since CUDA 5.5.
CUDBGResult (*CUDBGAPI_st::getGridStatus50) (uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)

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

Parameters

dev
- grid ID

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

Since CUDA 5.0.

Deprecated in CUDA 5.5.

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

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

Parameters

dev
- device index

sm
- SM index

wp
- warp index

tid
- the returned thread id

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.
3.8. Device Properties

CUDBGResult (*CUDBGAPI_st::getDeviceName)(uint32_t dev, char *buf, uint32_t sz)
Get the device name string.

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

Since CUDA 6.5.

See also:
getSMType
getDeviceType

CUDBGResult (*CUDBGAPI_st::getDeviceType)(uint32_t dev, char *buf, uint32_t sz)
Get the string description of the device.

Parameters

dev - device index
buf - the destination buffer
sz - the size of the buffer
Returns
CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALIDARGS,
CUDBG_ERROR_INVALIDDEVICE, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getSMTypetype

CUDBGResult (*CUDBGAPI_st::getNumDevices)(uint32_t *numDev)
Get the number of installed CUDA devices.

Parameters
numDev
  - the returned number of devices

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALIDARGS, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getNumSMs
getNumWarps
getNumLanes
getNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumLanes)(uint32_t dev, uint32_t *numLanes)
Get the number of lanes per warp on the device.

Parameters
dev
  - device index
numLanes
  - the returned number of lanes
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
getNumDevices
getNumSMs
ggetNumWarps
ggetNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumPredicates) (uint32_t dev, uint32_t *numPredicates)
Get the number of predicate registers per lane on the device.

Parameters
\texttt{dev}  
- device index

\texttt{numPredicates}
- the returned number of predicate registers

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 6.5.

See also:
getNumDevices
getNumSMs
ggetNumWarps
ggetNumLanes
ggetNumRegisters
CUDBGResult (*CUDBGAPI_st::getNumRegisters) (uint32_t dev, uint32_t *numRegs)
Get the number of registers per lane on the device.

Parameters

dev
- device index

numRegs
- the returned number of registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:

gNumDevices
gNumSMs
gNumWarps
gNumLanes

CUDBGResult (*CUDBGAPI_st::getNumSMs) (uint32_t dev, uint32_t *numSMs)
Get the total number of SMs on the device.

Parameters

dev
- device index

numSMs
- the returned number of SMs

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
**getNumDevices**

**getNumWarps**

**getNumLanes**

**getNumRegisters**

**CUDBGResult**

(*CUDASt::getNumUniformPredicates) (uint32_t
dev, uint32_t *numPredicates)

Get the number of uniform predicate registers per warp on the device.

**Parameters**

- **dev**
  - device index
- **numPredicates**
  - the returned number of uniform predicate registers

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 10.0.

**See also:**

**getNumUniformPredicates**

**CUDBGResult**

(*CUDASt::getNumUniformRegisters) (uint32_t
dev, uint32_t *numRegs)

Get the number of uniform registers per warp on the device.

**Parameters**

- **dev**
  - device index
- **numRegs**
  - the returned number of uniform registers
Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 10.0.

See also:

getNumRegisters

CUDBGResult (*CUDBGAPI_st::getNumWarps) (uint32_t dev, uint32_t *numWarps)
Get the number of warps per SM on the device.

Parameters

dev
- device index
numWarps
- the returned number of warps

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:

getNumDevices
getNumSMs
getNumLanes
getNumRegisters

CUDBGResult (*CUDBGAPI_st::getSmType) (uint32_t dev, char *buf, uint32_t sz)
Get the SM type of the device.

Parameters

dev
- device index
buf
- the destination buffer

sz
- the size of the buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
getDeviceType

3.9. DWARF Utilities

CUDBGResult (*CUDBGAPI_st::disassemble) (uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)
Disassemble instruction at instruction address.

Parameters

dev
- device index

addr
- instruction address

instSize
- instruction size (32 or 64 bits)

buf
- disassembled instruction buffer

sz
- disassembled instruction buffer size

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.
CUDBGResult (*CUDBGAPI_st::getElfImageByHandle) (uint32_t devId, uint64_t handle, CUDBGElfImageType type, void *elfImage, uint64_t size)
Get the relocated or non-relocated ELF image for the given handle on the given device.

Parameters

devId
- device index
handle
- elf image handle
type
- type of the requested ELF image
elfImage
- pointer to the ELF image
size

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

The handle is provided in the ELF Image Loaded notification event.

Since CUDA 6.0.

CUDBGResult (*CUDBGAPI_st::getHostAddrFromDeviceAddr) (uint32_t dev, uint64_t device_addr, uint64_t *host_addr)
given a device virtual address, return a corresponding system memory virtual address.

Parameters

dev
- device index
device_addr
- device memory address
host_addr
- returned system memory address
Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_CONTEXT, CUDBG_ERROR_INVALID_MEMORY_SEGMENT

Since CUDA 4.1.

See also:
readGenericMemory
writeGenericMemory

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

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

Parameters

pc
- Program counter
reg
- virtual register index
buf
- physical register name(s)
sz
- the physical register name buffer size
numPhysRegs
- number of physical register names returned
regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.
Deprecated in CUDA 3.1.
CUDBGResult
(*CUDBGAPI_st::getPhysicalRegister40) [uint32_t
dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg,
uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs,
CUDBGRegClass *regClass]

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

Parameters

dev
- device index

sm
- SM index

wp
- warp indx

pc
- Program counter

reg
- virtual register index

buf
- physical register name(s)

sz
- the physical register name buffer size

numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNKNOWN_FUNCTION,
CUDBG_ERROR_UNKNOWN

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

Since CUDA 3.1.

Deprecated in CUDA 4.1.
CUDBGResult (*CUDBGAPI_st::isDeviceCodeAddress) (uintptr_t addr, bool *isDeviceAddress)
Determines whether a virtual address resides within device code.

Parameters
addr
    - virtual address
isDeviceAddress
    - true if address resides within device code

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.

CUDBGResult (*CUDBGAPI_st::isDeviceCodeAddress55) (uintptr_t addr, bool *isDeviceAddress)
Determines whether a virtual address resides within device code. This API is strongly deprecated. Use CUDBGAPI_st::isDeviceCodeAddress instead.

Parameters
addr
    - virtual address
isDeviceAddress
    - true if address resides within device code

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.
Deprecated in CUDA 6.0
**CUDBGResult**

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

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

**Parameters**

- **symName**
  - symbol name
- **symFound**
  - set to true if the symbol is found
- **symAddr**
  - the symbol virtual address if found

**Returns**

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.

### 3.10. Events

One of those events will create a **CUDBGEvent**:

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

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

Upon notification the debugger is responsible for handling the CUDBGEvents in the event queue by using **CUDBGAPI_st::getNextEvent()**, and for acknowledging the debugger API that the event has been handled by calling **CUDBGAPI_st::acknowledgeEvent()**. In the case of an event raised by the device itself, such as a breakpoint being hit, the event queue will be empty. It is the responsibility of the debugger to inspect the hardware any time a **CUDBGEvent** is received.
Example:

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

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

**struct CUDBGEvent**

Event information container.

**struct CUDBGEventCallbackData**

Event information passed to callback set with setNotifyNewEventCallback function.

**struct CUDBGEventCallbackData40**

Event information passed to callback set with setNotifyNewEventCallback function.

**enum CUDBGEventKind**

CUDA Kernel Events.

**Values**

- `CUDBG_EVENT_INVALID = 0x000`
  - Invalid event.
- `CUDBG_EVENT_ELF_IMAGE_LOADED = 0x001`
  - The ELF image for a CUDA source module is available.
- `CUDBG_EVENT_KERNEL_READY = 0x002`
  - A CUDA kernel is about to be launched.
- `CUDBG_EVENT_KERNEL_FINISHED = 0x003`
  - A CUDA kernel has terminated.
- `CUDBG_EVENT_INTERNAL_ERROR = 0x004`
  - An internal error occur. The debugging framework may be unstable.
CUDBG_EVENT_CTX_PUSH = 0x005
A CUDA context was pushed.

CUDBG_EVENT_CTX_POP = 0x006
A CUDA CTX was popped.

CUDBG_EVENT_CTX_CREATE = 0x007
A CUDA CTX was created.

CUDBG_EVENT_CTX_DESTROY = 0x008
A CUDA context was destroyed.

CUDBG_EVENT_TIMEOUT = 0x009
An timeout event is sent at regular interval. This event can safely ignored.

CUDBG_EVENT_ATTACH_COMPLETE = 0x00a
The attach process has completed and debugging of device code may start.

CUDBG_EVENT_DETACH_COMPLETE = 0x00b
CUDBG_EVENT_ELF_IMAGE_UNLOADED = 0x00c
CUDBG_EVENT_FUNCTIONS_LOADED = 0x00d

typedef (*CUDBGNotifyNewEventCallback) (CUDBGEventCallbackData* data)
function type of the function called to notify debugger of the presence of a new event in the event queue.

typedef (*CUDBGNotifyNewEventCallback31) (void* data)
function type of the function called to notify debugger of the presence of a new event in the event queue. 
Deprecated in CUDA 3.2.

CUDBGResult (*CUDBGAPI_st::acknowledgeEvent30) (CUDBGEvent30 *event)
Inform the debugger API that the event has been processed.

Parameters

event
- pointer to the event that has been processed

Returns
CUDBG_SUCCESS
Since CUDA 3.0.
Deprecated in CUDA 3.1.
CUDBGResult
(*CUDBGAPI_st::acknowledgeEvents42) ()
Inform the debugger API that synchronous events have been processed.

Returns
CUDBG_SUCCESS
Since CUDA 3.1.
Deprecated in CUDA 5.0.

CUDBGResult
(*CUDBGAPI_st::acknowledgeSyncEvents) ()
Inform the debugger API that synchronous events have been processed.

Returns
CUDBG_SUCCESS
Since CUDA 5.0.

CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent50) (CUDBGEvent50 *event)
Copies the next available event in the asynchronous event queue into ‘event’ and removes it from the queue. The asynchronous event queue is held separate from the normal event queue, and does not require acknowledgement from the debug client.

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 5.0.
Deprecated in CUDA 5.5.
**CUDBGResult (*CUDBGAPI_st::getNextAsyncEvent55) (CUDBGEvent55 *event)**

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

**Parameters**

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

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.5.

**CUDBGResult (*CUDBGAPI_st::getNextEvent) (CUDBGEventQueueType type, CUDBGEvent *event)**

Copies the next available event into ‘event’ and removes it from the queue.

**Parameters**

- **type**
  - application event queue type
- **event**
  - pointer to an event container where to copy the event parameters

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 6.0.

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

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

**Parameters**

- **event**
  - pointer to an event container where to copy the event parameters
Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 3.0.
Deprecated in CUDA 3.1.

CUDBGResult (*CUDBGAPI_st::getNextEvent32) (CUDBGEvent32 *event)
Copies the next available event in the event queue into ‘event’ and removes it from the queue.

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 3.1.
Deprecated in CUDA 4.0

CUDBGResult (*CUDBGAPI_st::getNextEvent42) (CUDBGEvent42 *event)
Copies the next available event in the event queue into ‘event’ and removes it from the queue.

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 4.0.
Deprecated in CUDA 5.0
CUDBGResult (*CUDBGAPI_st::getNextSyncEvent50) (CUDBGEvent50 *event)

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

Returns
- CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.0.
Deprecated in CUDA 5.5.

CUDBGResult (*CUDBGAPI_st::getNextSyncEvent55) (CUDBGEvent55 *event)

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

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

Returns
- CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.5.

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

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

Parameters
- callback - the callback function
Returns
CUDBG_SUCCESS
Since CUDA 4.1.

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

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

Parameters

- **callback**
  - the callback function

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

Returns
CUDBG_SUCCESS
Since CUDA 3.0.
**Deprecated** in CUDA 3.2.

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

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

Parameters

- **callback**
  - the callback function

Returns
CUDBG_SUCCESS
Since CUDA 3.2.
 Deprecated in CUDA 4.1.
Chapter 4. Data Structures

Here are the data structures with brief descriptions:

- **cudbgGetAPI**
  The CUDA debugger API routines

- **CUDBGEvent**
  Event information container

- **CUDBGEvent::CUDBGEvent::cases_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::internalError_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st**

- **CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st**

- **CUDBGEventCallbackData**
  Event information passed to callback set with setNotifyNewEventCallback function

- **CUDBGEventCallbackData40**
  Event information passed to callback set with setNotifyNewEventCallback function

- **CUDBGGridInfo**
  Grid info

4.1. **CUDBGAPI_st Struct Reference**

The CUDA debugger API routines.
CUDBGResult (*acknowledgeEvent30) (CUDBGEvent30 *event)
Inform the debugger API that the event has been processed.

Parameters
event
- pointer to the event that has been processed

Returns
CUDBG_SUCCESS
Since CUDA 3.0.
Deprecated in CUDA 3.1.

CUDBGResult (*acknowledgeEvents42) ()
Inform the debugger API that synchronous events have been processed.

Returns
CUDBG_SUCCESS
Since CUDA 3.1.
Deprecated in CUDA 5.0.

CUDBGResult (*acknowledgeSyncEvents) ()
Inform the debugger API that synchronous events have been processed.

Returns
CUDBG_SUCCESS
Since CUDA 5.0.

CUDBGResult (*clearAttachState) ()
Clear attach-specific state prior to detach.

Returns
CUDBG_SUCCESS
Since CUDA 5.0.
**CUDBGResult (**disassemble**) (uint32_t dev, uint64_t addr, uint32_t *instSize, char *buf, uint32_t sz)**

Disassemble instruction at instruction address.

**Parameters**

- **dev**  
  - device index
- **addr**  
  - instruction address
- **instSize**  
  - instruction size (32 or 64 bits)
- **buf**  
  - disassembled instruction buffer
- **sz**  
  - disassembled instruction buffer size

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.

**CUDBGResult (**finalize**) ()**

Finalize the API and free all memory.

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_COMMUNICATION_FAILURE, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.

**See also:**

initialize
CUDA BRG (CUDA Debugger API)

1. **getAdjustedCodeAddress**

   ```c
   CUDA BRG (*getAdjustedCodeAddress) (uint32_t devId, uint64_t address, uint64_t *adjustedAddress, CUDA BRG AdjAddrAction adjAction)
   ```

   The client must call this function before inserting a breakpoint, or when the previous or next code address is needed. Returns the adjusted code address for a given code address for a given device.

   **Parameters**
   - **devId**: the device index
   - **address**: the address
   - **adjustedAddress**: the adjusted address
   - **adjAction**: whether the adjusted next, previous or current address is needed

   **Returns**
   - CUDA BRG_SUCCESS, CUDA BRG_ERROR_UNINITIALIZED, CUDA BRG_ERROR_INVALID_ADDRESS, CUDA BRG_ERROR_INVALID_DEVICE

   Since CUDA 5.5.

   **See also:**
   - unsetBreakpoint

2. **getBlockDim**

   ```c
   CUDA BRG (*getBlockDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockDim)
   ```

   Get the number of threads in the given block.

   **Parameters**
   - **dev**: device index
   - **sm**: SM index
   - **wp**: warp index
   - **blockDim**: the returned number of threads in the block
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getGridDim

CUDBGResult (*getClusterDim) (uint32_t dev, uint64_t gridId64, CuDim3 *clusterDim)
Get the number of blocks in the given cluster.

Parameters
dev
  - device index
gridId64
  - grid ID
clusterDim
  - the returned number of blocks in the cluster

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED
Since CUDA 12.0.

See also:
getBlockDim
getGridDim

CUDBGResult (*getDeviceName) (uint32_t dev, char *buf, uint32_t sz)
Get the device name string.

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

Since CUDA 6.5.

**See also:**

getSMType

getDeviceType

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

Get PCI bus and device ids associated with device devId.

**Parameters**

**devId**
- the cuda device id

**pciBusId**
- pointer where corresponding PCI BUS ID would be stored

**pciDevId**
- pointer where corresponding PCI DEVICE ID would be stored

**Returns**

CUDBG_ERROR_INVALID_ARGS, CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE

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

Get the string description of the device.

**Parameters**

**dev**
- device index

**buf**
- the destination buffer
sz
- the size of the buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_BUFFER_TOO_SMALL, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getSMType

CUDBGResult (*getElfImage) (uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint64_t *size)
Get the relocated or non-relocated ELF image and size for the grid on the given device.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
relocated
- set to true to specify the relocated ELF image, false otherwise
*elfImage
- pointer to the ELF image
size
- size of the ELF image (64 bits)

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED
Since CUDA 4.0.
Data Structures

CUDA Debugger API

TRM-06710-001_vRelease Version | 104

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

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

Parameters

dev
   - device index

sm
   - SM index

wp
   - warp index

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

*elfImage
   - pointer to the ELF image

size
   - size of the ELF image (32 bits)

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

Deprecated in CUDA 4.0.

-CUDBGResult (*getElfImageByHandle) (uint32_t devId, uint64_t handle, CUDBGElfImageType type, void *elfImage, uint64_t size)

Get the relocated or non-relocated ELF image for the given handle on the given device.

Parameters

devId
   - device index

handle
   - elf image handle

type
   - type of the requested ELF image
elfImage
- pointer to the ELF image

size

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

The handle is provided in the ELF Image Loaded notification event.

Since CUDA 6.0.

CUDBGResult (*getGridAttribute) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttribute attr, uint64_t *value)
Get the value of a grid attribute.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

attr
- the attribute

value
- the returned value of the attribute

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.1.
CUDBGResult (*getGridAttributes) (uint32_t dev, uint32_t sm, uint32_t wp, CUDBGAttributeValuePair *pairs, uint32_t numPairs)
Get several grid attribute values in a single API call.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

pairs
- array of attribute/value pairs

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

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INVALID_ATTRIBUTE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.1.

CUDBGResult (*getGridDim) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *gridDim)
Get the number of blocks in the given grid.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

gridDim
- the returned number of blocks in the grid
Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 4.0.

See also:

getBlockDim

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

Get the number of blocks in the given grid.

Parameters

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **gridDim**
  - the returned number of blocks in the grid

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

Deprecated in CUDA 4.0.

See also:

getBlockDim
CUDBGResult (*getGridInfo) (uint32_t dev, uint64_t gridId64, CUDBGGridInfo *gridInfo)
Get information about the specified grid. If the context of the grid has already been destroyed, the function will return CUDBG_ERROR_INVALID_GRID, although the grid id is correct.

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

Returns
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_GRID
- CUDBG_SUCCESS
Since CUDA 12.0.

CUDBGResult (*getGridInfo55) (uint32_t dev, uint64_t gridId64, CUDBGGridInfo55 *gridInfo)
Get information about the specified grid. If the context of the grid has already been destroyed, the function will return CUDBG_ERROR_INVALID_GRID, although the grid id is correct.

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

Returns
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_GRID
- CUDBG_SUCCESS
Since CUDA 5.5.
CUDBGResult (*getGridStatus) (uint32_t dev, uint64_t gridId64, CUDBGGridStatus *status)
Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

dev
gridId64
   - 64-bit grid ID
status
   - enum indicating whether the grid status is INVALID, PENDING, ACTIVE, SLEEPING, TERMINATED or UNDETERMINED

Returns
CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Since CUDA 5.5.

CUDBGResult (*getGridStatus50) (uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)
Check whether the grid corresponding to the given gridId is still present on the device.

Parameters

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

Returns
CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_INTERNAL

Since CUDA 5.0.
Deprecated in CUDA 5.5.
**CUDA Debugger API**

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

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

**Parameters**

- **dev**
  - device index
- **device_addr**
  - device memory address
- **host_addr**
  - returned system memory address

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_CONTEXT, CUDBG_ERROR_INVALID_MEMORY_SEGMENT

Since CUDA 4.1.

**See also:**

readGenericMemory
writeGenericMemory

**CUDBGResult (*getLoadedFunctionInfo) (uint32_t devId, uint64_t handle, CUDBGLoadedFunctionInfo *info, uint32_t numEntries)**

Get the section number and address of loaded functions for a given module.

**Parameters**

- **devId**
- **handle**
  - ELF/cubin image handle
- **info**
- **numEntries**
  - number of function load entries to read
CUDBGResult (*getManagedMemoryRegionInfo) (uint64_t startAddress, CUDBGMemoryInfo *memoryInfo, uint32_t memoryInfo_size, uint32_t *numEntries)

Returns a sorted list of managed memory regions. The sorted list of memory regions starts from a region containing the specified starting address. If the starting address is set to 0, a sorted list of managed memory regions is returned which starts from the managed memory region with the lowest start address.

Parameters

startAddress
- The address that the first region in the list must contain. If the starting address is set to 0, the list of managed memory regions returned starts from the managed memory region with the lowest start address.

memoryInfo
- Client-allocated array of memory region records of type CUDBGMemoryInfo.

memoryInfo_size
- Number of records of type CUDBGMemoryInfo that memoryInfo can hold.

numEntries
- Pointer to a client-allocated variable holding the number of valid entries returned in memoryInfo. Valid entries are contiguous and start from memoryInfo[0].

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INTERNAL

Since CUDA 6.0.
**CUDBGResult (*getNextAsyncEvent50) (CUDBGEvent50 *event)**

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

**Parameters**

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

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.0.

**Deprecated** in CUDA 5.5.

**CUDBGResult (*getNextAsyncEvent55) (CUDBGEvent55 *event)**

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

**Parameters**

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

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.5.

**CUDBGResult (*getNextEvent) (CUDBGEventQueueType type, CUDBGEvent *event)**

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

**Parameters**

- **type**
  - application event queue type
event
- pointer to an event container where to copy the event parameters

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 6.0.

CUDBGResult (*getNextEvent30) (CUDBGEvent30 *event)
Copies the next available event in the event queue into ‘event’ and removes it from the queue.

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 3.0.
Deprecated in CUDA 3.1.

CUDBGResult (*getNextEvent32) (CUDBGEvent32 *event)
Copies the next available event in the event queue into ‘event’ and removes it from the queue.

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 3.1.
Deprecated in CUDA 4.0
CUDBGResult (*getNextEvent42) (CUDBGEvent42 *event)
Copies the next available event in the event queue into `event` and removes it from the queue.

Parameters

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 4.0.
Deprecated in CUDA 5.0.

CUDBGResult (*getNextSyncEvent50) (CUDBGEvent50 *event)

Parameters

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

Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS

Since CUDA 5.0.
Deprecated in CUDA 5.5.

CUDBGResult (*getNextSyncEvent55) (CUDBGEvent55 *event)
Copies the next available event in the synchronous event queue into `event` and removes it from the queue.

Parameters

- **event**
  - pointer to an event container where to copy the event parameters
Returns
CUDBG_SUCCESS, CUDBG_ERROR_NO_EVENT_AVAILABLE, CUDBG_ERROR_INVALID_ARGS
Since CUDA 5.5.

`CUDBGResult (*getNumDevices) (uint32_t *numDev)`
Get the number of installed CUDA devices.

Parameters
`numDev`
- the returned number of devices

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
`getNumSMs`
`getNumWarps`
`getNumLanes`
`getNumRegisters`

`CUDBGResult (*getNumLanes) (uint32_t dev, uint32_t *numLanes)`
Get the number of lanes per warp on the device.

Parameters
`dev`
- device index
`numLanes`
- the returned number of lanes

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.
**See also:**
getNumDevices  
getNumSMs  
getNumWarps  
getNumRegisters

**CUDADebugger** (*getNumPredicates*) *(uint32_t dev, uint32_t *numPredicates)*
Get the number of predicate registers per lane on the device.

**Parameters**

- **dev** - device index  
- **numPredicates** - the returned number of predicate registers

**Returns**

CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_UNINITIALIZED

Since CUDA 6.5.

**See also:**
getNumDevices  
getNumSMs  
getNumWarps  
getNumLanes  
getNumRegisters

**CUDADebugger** (*getNumRegisters*) *(uint32_t dev, uint32_t *numRegs)*
Get the number of registers per lane on the device.

**Parameters**

- **dev** - device index
numRegs
   - the returned number of registers

Returns
CUDA_DEBUG_SUCCESS, CUDA_DEBUG_ERROR_INVALID_ARGS, CUDA_DEBUG_ERROR_INVALID_DEVICE, CUDA_DEBUG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getNumDevices
getNumSMs
getNumWarps
getNumLanes

CUDBGResult (*getNumSMs) (uint32_t dev, uint32_t *numSMs)
Get the total number of SMs on the device.

Parameters
dev
   - device index
numSMs
   - the returned number of SMs

Returns
CUDA_DEBUG_SUCCESS, CUDA_DEBUG_ERROR_INVALID_ARGS, CUDA_DEBUG_ERROR_INVALID_DEVICE, CUDA_DEBUG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
getNumDevices
getNumWarps
getNumLanes
getNumRegisters
**CUDA**

**CUDA Debugger API**

---

**CUDA Debugger Result** (*getNumUniformPredicates*) (uint32_t dev, uint32_t *numPredicates)

Get the number of uniform predicate registers per warp on the device.

**Parameters**

- **dev**
  - device index

- **numPredicates**
  - the returned number of uniform predicate registers

**Returns**

- CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_UNINITIALIZED

Since CUDA 10.0.

**See also:**

- getNumUniformPredicates

---

**CUDA Debugger Result** (*getNumUniformRegisters*) (uint32_t dev, uint32_t *numRegs)

Get the number of uniform registers per warp on the device.

**Parameters**

- **dev**
  - device index

- **numRegs**
  - the returned number of uniform registers

**Returns**

- CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_UNINITIALIZED

Since CUDA 10.0.

**See also:**

- getNumRegisters
CUDBGResult (*getNumWarps) (uint32_t dev, uint32_t *numWarps)
Get the number of warps per SM on the device.

Parameters

- **dev**
  - device index

- **numWarps**
  - the returned number of warps

Returns

- CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:

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

CUDBGResult (*getPhysicalRegister30) (uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)
Get the physical register number[s] assigned to a virtual register name 'reg' at a given PC, if 'reg' is live at that PC.

Parameters

- **pc**
  - Program counter

- **reg**
  - virtual register index

- **buf**
  - physical register name(s)

- **sz**
  - the physical register name buffer size
numPhysRegs
- number of physical register names returned

regClass
- the class of the physical registers

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.
Deprecated in CUDA 3.1.

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

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

Parameters
dev
- device index
sm
- SM index
wp
- warp indx
pc
- Program counter
reg
- virtual register index
buf
- physical register name(s)
sz
- the physical register name buffer size
numPhysRegs
- number of physical register names returned
regClass
- the class of the physical registers
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNKNOWN_FUNCTION, CUDBG_ERROR_UNKNOWN

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

Since CUDA 3.1.
Deprecated in CUDA 4.1.

CUDBGResult (*getSmType) (uint32_t dev, char *buf, uint32_t sz)
Get the SM type of the device.

Parameters

\texttt{dev} \\
- device index

\texttt{buf} \\
- the destination buffer

\texttt{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

Since CUDA 3.0.

See also:
getDeviceType
CUDBGResult (*getTID) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *tid)

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

Parameters

dev
- device index

sm
- SM index

wp
- warp index

tid
- the returned thread id

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_GRID, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

CUDBGResult (*initialize) ()

Initialize the API.

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNKNOWN

Since CUDA 3.0.

See also:

finalize

CUDBGResult (*initializeAttachStub) ()

Initialize the attach stub.

Returns

CUDBG_SUCCESS

Since CUDA 5.0.
CUDBGResult (*isDeviceCodeAddress) (uintptr_t addr, bool *isDeviceAddress)
Determines whether a virtual address resides within device code.

Parameters

addr
- virtual address

isDeviceAddress
- true if address resides within device code

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.

CUDBGResult (*isDeviceCodeAddress55) (uintptr_t addr, bool *isDeviceAddress)
Determines whether a virtual address resides within device code. This API is strongly deprecated. Use CUDBGAPI_st::isDeviceCodeAddress instead.

Parameters

addr
- virtual address

isDeviceAddress
- true if address resides within device code

Returns

CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS

Since CUDA 3.0.

Deprecated in CUDA 6.0
CUDBGResult (*lookupDeviceCodeSymbol)(char *symName, bool *symFound, uintptr_t *symAddr)
Determines whether a symbol represents a function in device code and returns its virtual address.

Parameters
symName
- symbol name
symFound
- set to true if the symbol is found
symAddr
- the symbol virtual address if found

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_UNINITIALIZED, CUDBG_SUCCESS
Since CUDA 3.0.

CUDBGResult (*memcheckReadErrorAddress)(uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *address, ptxStorageKind *storage)
Get the address that memcheck detected an error on.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
address
- returned address detected by memcheck
storage
- returned address class of address

Returns
CUDBG_ERROR_NOT_SUPPORTED,
Since CUDA 5.0.

**`CUDAResult (*readActiveLanes) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *activeLanesMask)`**

Reads the bitmask of active lanes on a valid warp.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **activeLanesMask**
  - the returned bitmask of active lanes

**Returns**

- `CUDA_SUCCESS`, `CUDA_ERROR_INVALID_ARGS`, `CUDA_ERROR_INVALID_DEVICE`, `CUDA_ERROR_INVALID_SM`, `CUDA_ERROR_INVALID_WARP`, `CUDA_ERROR_UNINITIALIZED`

Since CUDA 3.0.

**See also:**

- `readGridId`
- `readBlockIdx`
- `readThreadIdx`
- `readBrokenWarps`
- `readValidWarps`
- `readValidLanes`

**`CUDAResult (*readBlockIdx) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *blockIdx)`**

Reads the CUDA block index running on a valid warp.

**Parameters**

- **dev**
  - device index
sm
  - SM index
wp
  - warp index
blockIdx
  - the returned CUDA block index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 4.0.

See also:
readGridId
readThreadIdx
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

CUDBGResult (*readBlockIdx32) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)
Reads the two-dimensional CUDA block index running on a valid warp.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
blockIdx
  - the returned CUDA block index
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.
Deprecated in CUDA 4.0.

See also:
readGridId
readThreadId
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

CUDBGResult (*readBrokenWarps) (uint32_t dev, uint32_t sm, uint64_t *brokenWarpsMask)
Reads the bitmask of warps that are at a breakpoint on a given SM.

Parameters
dev
- device index
sm
- SM index
brokenWarpsMask
- the returned bitmask of broken warps

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
readGridId
readBlockIdx
**readThreadIdx**

**readValidWarps**

**readValidLanes**

**readActiveLanes**

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

Reads the call depth [number of calls] for a given lane.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **ln**
  - lane index
- **depth**
  - the returned call depth

**Returns**

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

Since CUDA 4.0.

**See also:**

**readReturnAddress**

**readVirtualReturnAddress**

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

Reads the call depth [number of calls] for a given warp.

**Parameters**

- **dev**
  - device index
**sm**
- SM index

**wp**
- warp index

**depth**
- the returned call depth

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.1.

**Deprecated** in CUDA 4.0.

**See also:**

readReturnAddress32
readVirtualReturnAddress32

**CUDBGResult (*readCCRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *val)**

 Reads the hardware CC register.

**Parameters**

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**ln**
- lane 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

Since CUDA 6.5.
See also:
readCodeMemory
readConstMemory
readGenericMemory
readGlobalMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
readPredicates

CUDBGResult (*readClusterIdx) (uint32_t dev, uint32_t sm, uint32_t wp, CuDim3 *clusterIdx)
Reads the CUDA cluster index running on a valid warp.

Parameters

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**clusterIdx**
- the returned CUDA cluster index

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 12.0.

See also:
readGridId
**CUDA Debugger API**

### readThreadIdx

**readBlockIdx**

**readBrokenWarps**

**readValidWarps**

**readValidLanes**

**readActiveLanes**

---

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

Reads content at address in the code memory segment.

**Parameters**

- **dev**
  - device index
- **addr**
  - memory address
- **buf**
  - buffer
- **sz**
  - size of the buffer

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Since CUDA 3.0.

**See also:**

- readConstMemory
- readGenericMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readLocalMemory
- readRegister
- readPC
CUDBGResult (*readConstMemory) (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
Reads content at address in the constant memory segment.

Parameters

- **dev** - device index
- **addr** - memory address
- **buf** - buffer
- **sz** - size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED

Since CUDA 3.0.

See also:

- readCodeMemory
- readGenericMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readLocalMemory
- readRegister
- readPC

CUDBGResult (*readDeviceExceptionState) (uint32_t devId, uint64_t *mask, uint32_t numWords)
Get the exception state of the SMs on the device.

Parameters

- **devId** - the cuda device id
mask
  - Arbitrarily sized bit field containing a 1 at (1 << i) if SM i hit an exception

numWords

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE
Since CUDA 9.0

See also:
getNumSMs

CUDBGResult (*readDeviceExceptionState80) (uint32_t devId, uint64_t *exceptionSMMask)
Get the exception state of the SMs on the device.

Parameters

devId
  - the cuda device id

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

Returns
CUDBG_ERROR_INVALID_ARGS, CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE
Since CUDA 5.5

CUDBGResult (*readErrorPC) (uint32_t devId, uint32_t sm, uint32_t wp, uint64_t *errorPC, bool *errorPCValid)
Get the hardware reported error PC if it exists.

Parameters

devId
  - the device index

sm
  - the SM index

wp

errorPC
  - PC of the exception
errorPCValid
- boolean to indicate that the returned error PC is valid

Returns
CUDBG_SUCCESS CUBDG_ERROR_UNINITIALIZED CUBDG_ERROR_INVALID_DEVICE
CUBDG_ERROR_INVALID_SM CUBDG_ERROR_INVALID_WARP
CUBDG_ERROR_INVALID_ARGS CUBDG_ERROR_UNKNOWN_FUNCTION

Since CUDA 6.0

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

Reads content at an address in the generic address space. This function determines if the given address falls into the local, shared, or global memory window. It then accesses memory taking into account the hardware co-ordinates provided as inputs.

Parameters
    dev
    - device index

    sm
    - SM index

    wp
    - warp index

    ln
    - lane index

    addr
    - memory address

    buf
    - buffer

    sz

Returns
CUDBG_SUCCESS, CUBDG_ERROR_INVALID_ARGS, CUBDG_ERROR_INVALID_DEVICE, CUBDG_ERROR_INVALID_LANE, CUBDG_ERROR_INVALID_SM, CUBDG_ERROR_INVALID_WARP, CUBDG_ERROR_UNINITIALIZED, CUBDG_ERROR_MEMORY_MAPPING_FAILED, CUBDG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM

Since CUDA 6.0.

See also:
CUDBGResult (*readGlobalMemory) (uint64_t addr, void *buf, uint32_t sz)

Reads content at an address in the global address space. If the address is valid on more than one device and one of those devices does not support UVA, an error is returned.

Parameters

- **addr** - memory address
- **buf** - buffer
- **sz**

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_INVALID_MEMORY_ACCESS, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM, CUDBG_ERROR_AMBIGUOUS_MEMORY_ADDRESS_

Since CUDA 6.0.

See also:

- readCodeMemory
- readConstMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readLocalMemory
- readRegister
- readPC
**readRegister**

**readPC**

**CUDADebugger**

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

Reads content at address in the global memory segment.

**Parameters**

- **dev**
  - device index
- **addr**
  - memory address
- **buf**
  - buffer
- **sz**
  - size of the buffer

**Returns**

CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_UNINITIALIZED, CUDA_ERROR_MEMORY_MAPPING_FAILED

Since CUDA 3.0.

**Deprecated** in CUDA 3.2.

**See also:**

readCodeMemory
readConstMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
CUDBGResult (*readGlobalMemory55) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

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

Parameters

dev - device index
sm - SM index
wp - warp index
ln - lane index
addr - memory address
buf - buffer
sz - size of the buffer

Returns

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

Since CUDA 3.2.
Deprecated in CUDA 6.0.

See also:
readCodeMemory
readConstMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
**readRegister**
**readPC**

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

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

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **gridId64**

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP, CUDBG_ERROR_UNINITIALIZED

Since CUDA 5.5.

**See also:**

- **readBlockIdx**
- **readThreadIdx**
- **readBrokenWarps**
- **readValidWarps**
- **readValidLanes**
- **readActiveLanes**

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

Reads the CUDA grid index running on a valid warp.

**Parameters**

- **dev**
  - device index
sm
- SM index
wp
- warp index
gridId
- the returned CUDA grid index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.
Deprecated in CUDA 5.5.

See also:
readBlockIdx
readThreadIdx
readBrokenWarps
readValidWarps
readValidLanes
readActiveLanes

CUDBGResult (*readLaneException) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln,
CUDBGException_t *exception)
Reads the exception type for a given lane.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
exception
- the returned exception type

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE,
CUDA_ERROR_INVALID_LANE, CUDA_ERROR_INVALID_SM,
CUDA_ERROR_INVALID_WARP, CUDA_ERROR_UNINITIALIZED

Since CUDA 3.1.

CUDBGResult (*readLaneStatus) (uint32_t dev,
uint32_t sm, uint32_t wp, uint32_t ln, bool *error)
Reads the status of the given lane. For specific error values, use readLaneException.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

error
- true if there is an error

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_ARGS, CUDA_ERROR_INVALID_DEVICE,
CUDA_ERROR_INVALID_LANE, CUDA_ERROR_INVALID_SM,
CUDA_ERROR_INVALID_WARP, CUDA_ERROR_UNINITIALIZED

Since CUDA 3.0.
CUDBGResult (*readLocalMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)

Reads content at address in the local memory segment.

Parameters

- **dev**
  - device index

- **sm**
  - SM index

- **wp**
  - warp index

- **ln**
  - lane index

- **addr**
  - memory address

- **buf**
  - buffer

- **sz**
  - size of the buffer

Returns

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

Since CUDA 3.0.

See also:

- readCodeMemory
- readConstMemory
- readGenericMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readRegister
- readPC
`CUDBGResult (*readParamMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, void *buf, uint32_t sz)`

Reads content at address in the param memory segment.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **addr**
  - memory address
- **buf**
  - buffer
- **sz**
  - size of the buffer

**Returns**

- `CUDBG_SUCCESS`
- `CUDBG_ERROR_INVALID_ARGS`
- `CUDBG_ERROR_INVALID_DEVICE`
- `CUDBG_ERROR_INVALID_SM`
- `CUDBG_ERROR_INVALID_WARP`
- `CUDBG_ERROR_UNINITIALIZED`
- `CUDBG_ERROR_MEMORY_MAPPING_FAILED`

Since CUDA 3.0.

**See also:**

- `readCodeMemory`
- `readConstMemory`
- `readGenericMemory`
- `readSharedMemory`
- `readTextureMemory`
- `readLocalMemory`
- `readRegister`
- `readPC`
CUDBGResult (*readPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)
Reads the PC on the given active lane.

Parameters

  *dev
  - device index
  *sm
  - SM index
  *wp
  - warp index
  *ln
  - lane index
  *pc
  - the returned PC

Returns

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

Since CUDA 3.0.

See also:

  readCodeMemory
  readConstMemory
  readGenericMemory
  readParamMemory
  readSharedMemory
  readTextureMemory
  readLocalMemory
  readRegister
  readVirtualPC
CUDBGResult (*readPinnedMemory) (uint64_t addr, void *buf, uint32_t sz)
Reads content at pinned address in system memory.

Parameters

addr
- system memory address

buf
- buffer

sz
- size of the buffer

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS,
CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.2.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readRegister
readPC
CUDBGResult (*readPredicates) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t predicates_size, uint32_t *predicates)

Reads content of hardware predicate registers.

Parameters

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **ln**
  - lane index
- **predicates_size**
  - number of predicate registers to read
- **predicates**
  - 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

Since CUDA 6.5.

See also:

- readCodeMemory
- readConstMemory
- readGenericMemory
- readGlobalMemory
- readParamMemory
- readSharedMemory
- readTextureMemory
- readLocalMemory
- readRegister
- readPC
CUDBGResult (*readRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t regno, uint32_t *val)

Reads content of a hardware register.

Parameters

device
  - 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

Since CUDA 3.0.

See also:

readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readPC
CUDBGResult (*readRegisterRange) (uint32_t devId, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t index, uint32_t registers_size, uint32_t *registers)

Reads content of a hardware range of hardware registers.

Parameters

devId
sm
    - SM index
wp
    - warp index
ln
    - lane index
index
    - index of the first register to read
registers_size
    - number of registers to read
registers
    - 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

Since CUDA 6.0.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readTextureMemory
readLocalMemory
readPC
readRegister
CUDBGResult (*readReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)

Reads the physical return address for a call level.

Parameters

dev
  - device index

sm
  - SM index

wp
  - warp index

ln
  - lane index

level
  - the specified call level

ra
  - the returned return address for level

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALIDDEVICE, 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

Since CUDA 4.0.

See also:

readCallDepth
readVirtualReturnAddress

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

Reads the physical return address for a call level.

Parameters

dev
  - device index
sm
  - SM index
wp
  - warp index
level
  - the specified call level
ra
  - the returned return address for level

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

Since CUDA 3.1.
[Deprecated in CUDA 4.0.]

See also:
readCallDepth32
readVirtualReturnAddress32

CUDBGResult (*readSharedMemory) (uint32_t dev,
uint32_t sm, uint32_t wp, uint64_t addr, void *buf,
uint32_t sz)
Reads content at address in the shared memory segment.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
addr
  - memory address
buf
  - buffer
sz
- size of the buffer

Returns
CUDA_DBG_SUCCESS, CUDA_DBG_ERROR_INVALID_ARGS, CUDA_DBG_ERROR_INVALID_DEVICE,
CUDA_DBG_ERROR_INVALID_SM, CUDA_DBG_ERROR_INVALID_WARP,
CUDA_DBG_ERROR_UNINITIALIZED, CUDA_DBG_ERROR_MEMORY_MAPPING_FAILED
Since CUDA 3.0.

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readLocalMemory
readTextureMemory
readRegister
readPC

CUDA_DBGResult (*readSyscallCallDepth)(uint32_t dev,
uint32_t sm, uint32_t wp, uint32_t ln, uint32_t *depth)
Reads the call depth of syscalls for a given lane.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
depth
- the returned call depth
Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_INVALID_LANE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 4.1.

See also:
readReturnAddress
readVirtualReturnAddress

CUDBGResult (*readTextureMemory) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t id, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)
This method is no longer supported since CUDA 12.0.

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_ERROR_NOT_SUPPORTED,

See also:
readCodeMemory
readConstMemory
readGenericMemory
readParamMemory
readSharedMemory
readLocalMemory
readRegister
readPC

CUDBGResult (*readTextureMemoryBindless) (uint32_t devId, uint32_t vsm, uint32_t wp, uint32_t texSymtabIndex, uint32_t dim, uint32_t *coords, void *buf, uint32_t sz)
This method is no longer supported since CUDA 12.0.

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_ERROR_NOT_SUPPORTED

See also:
readCodeMemory
CUDA Debugger API

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

Reads the CUDA thread index running on valid lane.

Parameters

- **dev**  
  - device index
- **sm**  
  - SM index
- **wp**  
  - warp index
- **ln**  
  - lane index
- **threadIdx**  
  - the returned CUDA thread index

Returns

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

Since CUDA 3.0.

See also:

- readGridIdx
- readBlockIdx
- readBrokenWarps
- readValidWarps
- readValidLanes
**readActiveLanes**

**CUDBGResult (*readUniformPredicates) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t predicates_size, uint32_t *predicates)**

Reads contents of uniform predicate registers.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **predicates_size**
  - number of predicate registers to read
- **predicates**
  - buffer

**Returns**

- **CUDBG_SUCCESS**
- **CUDBG_ERROR_INVALID_ARGS**
- **CUDBG_ERROR_INVALID_DEVICE**
- **CUDBG_ERROR_INVALID_SM**
- **CUDBG_ERROR_INVALID_WARP**
- **CUDBG_ERROR_UNINITIALIZED**

Since CUDA 10.0.

**See also:**

**readPredicates**

**CUDBGResult (*readUniformRegisterRange) (uint32_t devId, uint32_t sm, uint32_t wp, uint32_t regno, uint32_t registers_size, uint32_t *registers)**

Reads a range of uniform registers.

**Parameters**

- **devId**
- **sm**
  - SM index
wp
- warp index

regno
- starting index into uniform register file

registers_size
- number of bytes to read

registers
- pointer to buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 10.0.

See also:
readRegister

CUDBGResult (*readValidLanes) (uint32_t dev,
uint32_t sm, uint32_t wp, uint32_t *validLanesMask)
Reads the bitmask of valid lanes on a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

validLanesMask
- the returned bitmask of valid lanes

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
CUDA Debugger API

TRM-06710-001 _vRelease Version   |   156

readGridId
readBlockIdx
readThreadIdx
readBrokenWarps
readValidWarps
readActiveLanes

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

Reads the bitmask of valid warps on a given SM.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **validWarpsMask**
  - the returned bitmask of valid warps

**Returns**

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

**See also:**

readGridId
readBlockIdx
readThreadIdx
readBrokenWarps
readValidLanes
readActiveLanes
CUDBGResult (*readVirtualPC) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t *pc)
Reads the virtual PC on the given active lane.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

ln
- lane index

pc
- the returned PC

Returns

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

Since CUDA 3.0.

See also:

readPC

CUDBGResult (*readVirtualReturnAddress) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t level, uint64_t *ra)
Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index
ln
- lane index

level
- the specified call level

ra
- the returned virtual return address for level

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

Since CUDA 4.0.

See also:
readCallDepth
readReturnAddress

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

Reads the virtual return address for a call level.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

level
- the specified call level

ra
- the returned virtual return address for level

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

Since CUDA 3.1. 
Deprecated in CUDA 4.0.

See also:
readCallDepth32
readReturnAddress32

CUDBGResult (*readWarpState) (uint32_t dev, 
uint32_t sm, uint32_t wp, CUDBGWarpState *state)

Get state of a given warp.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

state
- pointer to structure that contains warp status

Returns

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

Since CUDA 12.0.
CUDBGResult (*readWarpState60) (uint32_t devId, uint32_t sm, uint32_t wp, CUDBGWarpState60 *state)

Get state of a given warp.

Parameters

devid

sm
- SM index

wp
- warp index

state
- pointer to structure that contains warp status

Returns

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

Since CUDA 6.0.

CUDBGResult (*requestCleanupOnDetach) (uint32_t appResumeFlag)

Request for cleanup of driver state when detaching.

Parameters

appResumeFlag
- value of CUDBG_RESUME_FOR_ATTACH_DETACH as read from the application’s process space.

Returns

CUDBG_SUCCESS CUDBG_ERROR_COMMUNICATION_FAILURE
CUDBG_ERROR_INVALID_ARGS CUDBG_ERROR_INTERNAL

Since CUDA 6.0.
CUDBGResult (*requestCleanupOnDetach55) ()
Request for cleanup of driver state when detaching.

Returns
CUDBG_SUCCESS CUDBG_ERROR_COMMUNICATION_FAILURE
CUDBG_ERROR_INVALID_ARGS CUDBG_ERROR_INTERNAL

Since CUDA 5.0.
Deprecated in CUDA 6.0

CUDBGResult (*resumeDevice) (uint32_t dev)
Resume a suspended CUDA device.

Parameters
dev
- device index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE,
CUDBG_ERROR_UNINITIALIZED

Since CUDA 3.0.

See also:
suspendDevice
singleStepWarp

CUDBGResult (*resumeWarpsUntilPC) (uint32_t devId, uint32_t sm, uint64_t warpMask, uint64_t virtPC)
Inserts a temporary breakpoint at the specified virtual PC, and resumes all warps in the specified bitmask on a given SM. As compared to CUDBGAPI_st::resumeDevice,
CUDBGAPI_st::resumeWarpsUntilPC provides finer-grain control by resuming a selected set of warps on the same SM. The main intended usage is to accelerate the single-stepping process when the target PC is known in advance. Instead of single-stepping each warp individually until the target PC is hit, the client can issue this API. When this API is used, errors within CUDA kernels will no longer be reported precisely. In the situation where resuming warps is not possible, this API will return
CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE. The client should then fall back to using
CUDBGAPI_st::singleStepWarp or CUDBGAPI_st::resumeDevice.

Parameters

- **devId**
  - device index
- **sm**
  - the SM index
- **warpMask**
  - the bitmask of warps to resume (1 = resume, 0 = do not resume)
- **virtPC**
  - the virtual PC where the temporary breakpoint will be inserted

Returns

CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP_MASK,
CUDBG_ERROR_WARP_RESUME_NOT_POSSIBLE, CUDBG_ERROR_UNINITIALIZED

Since CUDA 6.0.

See also:

- resumeDevice

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

Sets a breakpoint at the given instruction address for the given device. Before setting a
breakpoint, CUDBGAPI_st::getAdjustedCodeAddress should be called to get the adjusted
breakpoint address.

Parameters

- **dev**
  - the device index
- **addr**
  - instruction address

Returns

CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS,
CUDBG_ERROR_INVALID_DEVICE

Since CUDA 3.2.

See also:
**unsetBreakpoint**

**CUDBGResult (*setBreakpoint31) (uint64_t addr)**
Sets a breakpoint at the given instruction address.

**Parameters**
- **addr**
  - instruction address

**Returns**
- CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS

Since CUDA 3.0.

*Deprecated* in CUDA 3.2.

**See also:**
- unsetBreakpoint31

**CUDBGResult (*setKernelLaunchNotificationMode) (CUDBGKernelLaunchNotifyMode mode)**
Set the launch notification policy.

**Parameters**
- **mode**
  - mode to deliver kernel launch notifications in

**Returns**
- CUDBG_SUCCESS

Since CUDA 5.5.

**CUDBGResult (*setNotifyNewEventCallback) (CUDBGNotifyNewEventCallback callback)**
Provides the API with the function to call to notify the debugger of a new application or device event.

**Parameters**
- **callback**
  - the callback function
Returns
CUDBG_SUCCESS
Since CUDA 4.1.

CUDBGResult (*setNotifyNewEventCallback31)
(CUDBGNotifyNewEventCallback31 callback, void *data)
Provides the API with the function to call to notify the debugger of a new application or device event.

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

Returns
CUDBG_SUCCESS
Since CUDA 3.0.
Deprecated in CUDA 3.2.

CUDBGResult (*setNotifyNewEventCallback40)
(CUDBGNotifyNewEventCallback40 callback)
Provides the API with the function to call to notify the debugger of a new application or device event.

Parameters
callback
- the callback function

Returns
CUDBG_SUCCESS
Since CUDA 3.2.
Deprecated in CUDA 4.1.
CUDBGResult (*singleStepWarp) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t nsteps, uint64_t *warpMask)

Single step an individual warp nsteps times on a suspended CUDA device. Only the last instruction in a range should be a control flow instruction.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **nsteps**
  - number of single steps
- **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

Since CUDA 7.5.

**See also:**

- resumeDevice
- suspendDevice

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

Single step an individual warp on a suspended CUDA device.

**Parameters**

- **dev**
  - device index
- **sm**
  - SM index
wp
- warp index

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_INVALID_SM, CUDA_ERROR_INVALID_WARP, CUDA_ERROR_RUNNING_DEVICE, CUDA_ERROR_UNINITIALIZED, CUDA_ERROR_UNKNOWN, CUDA_ERROR_WARP_RESUME_NOT_POSSIBLE

Since CUDA 3.0.
Deprecated in CUDA 4.1.

See also:
resumeDevice
suspendDevice
singleStepWarp

CUDADEBUGResult (*singleStepWarp41) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t *warpMask)
Single step an individual warp on a suspended CUDA device.

Parameters
dev
- device index
sm
- SM index
wp
- warp index
warpMask
- the warps that have been single-stepped

Returns
CUDA_SUCCESS, CUDA_ERROR_INVALID_DEVICE, CUDA_ERROR_INVALID_SM, CUDA_ERROR_INVALID_WARP, CUDA_ERROR_RUNNING_DEVICE, CUDA_ERROR_UNINITIALIZED, CUDA_ERROR_UNKNOWN

Since CUDA 4.1.
Deprecated in CUDA 7.5.

See also:
resumeDevice
suspendDevice

CUDBGResult (*suspendDevice) (uint32_t dev)
Suspends a running CUDA device.

Parameters
  dev
  - device index

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_DEVICE, CUDBG_ERROR_RUNNING_DEVICE, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.

See also:
resumeDevice
singleStepWarp

CUDBGResult (*unsetBreakpoint) (uint32_t dev, uint64_t addr)
Unsets a breakpoint at the given instruction address for the given device.

Parameters
  dev
  - the device index
  addr
  - instruction address

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_INVALID_ADDRESS, CUDBG_ERROR_INVALID_DEVICE
Since CUDA 3.2.

See also:
setBreakpoint
CUDBGResult (*unsetBreakpoint31) (uint64_t addr)
Unsets a breakpoint at the given instruction address.

Parameters
addr
  - instruction address

Returns
CUDBG_SUCCESS, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.0.
Deprecated in CUDA 3.2.

See also:
setBreakpoint31

CUDBGResult (*writeCCRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t val)
Writes the hardware CC register.

Parameters
dev
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
val
  - value to write to the CC register

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
Since CUDA 6.5.

See also:
writeConstMemory
writeGenericMemory
writeGlobalMemory
writeParamMemory
writeSharedMemory
writeTextureMemory
writeLocalMemory
writeRegister
writePredicates

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

Writes content to an address in the generic address space. This function determines if the given address falls into the local, shared, or global memory window. It then accesses memory taking into account the hardware co-ordinates provided as inputs.

Parameters

dev
- device index
sm
- SM index
wp
- warp index
ln
- lane index
addr
- memory address
buf
- buffer
sz

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
Since CUDA 6.0.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister

CUDBGResult (*writeGlobalMemory) (uint64_t addr, const void *buf, uint32_t sz)
Writes content to an address in the global address space. If the address is valid on more than one device and one of those devices does not support UVA, an error is returned.

Parameters
addr
- memory address
buf
- buffer
sz

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALIDDEVICE, CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_INVALID_MEMORY_ACCESS, CUDBG_ERROR_ADDRESS_NOT_IN_DEVICE_MEM, CUDBG_ERROR_AMBIGUOUS_MEMORY_ADDRESS

Since CUDA 6.0.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister
CUDBGResult (*writeGlobalMemory31) (uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)
Writes content to address in the global memory segment.

Parameters

- **dev**
  - device index
- **addr**
  - memory address
- **buf**
  - buffer
- **sz**
  - size of the buffer

Returns

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

Since CUDA 3.0.

*Deprecated* in CUDA 3.2.

See also:

- writeParamMemory
- writeSharedMemory
- writeLocalMemory
- writeRegister

CUDBGResult (*writeGlobalMemory55) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)
Writes content to address in the global memory segment [entire 40-bit VA on Fermi+].

Parameters

- **dev**
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
addr
  - memory address
buf
  - buffer
sz
  - size of the buffer

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

Since CUDA 3.2.
Deprecated in CUDA 6.0.

See also:
writeParamMemory
writeSharedMemory
writeLocalMemory
writeRegister

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

Writes content to address in the local memory segment.

Parameters
dev
  - device index
sm
  - SM index
wp
- warp index

ln
- lane index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

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

Since CUDA 3.0.

See also:
writeGenericMemory
writeParamMemory
writeSharedMemory
writeRegister

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

Writes content to address in the param memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address
buf
  - buffer
sz
  - size of the buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED, CUDBG_ERROR_MEMORY_MAPPING_FAILED
Since CUDA 3.0.

See also:
writeGenericMemory
writeSharedMemory
writeLocalMemory
writeRegister

CUDBGResult (*writePinnedMemory) (uint64_t addr,
const void *buf, uint32_t sz)
Writes content to pinned address in system memory.

Parameters
addr
  - system memory address
buf
  - buffer
sz
  - size of the buffer

Returns
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS,
CUDBG_ERROR_MEMORY_MAPPING_FAILED, CUDBG_ERROR_UNINITIALIZED
Since CUDA 3.2.

See also:
readCodeMemory
readConstMemory
CUDBGResult (*writePredicates) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint32_t predicates_size, const uint32_t *predicates)

Writes content to hardware predicate registers.

Parameters

dev
  - device index
sm
  - SM index
wp
  - warp index
ln
  - lane index
predicates_size
  - number of predicate registers to write
predicates
  - 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

Since CUDA 6.5.

See also:

writeConstMemory
writeGenericMemory
writeGlobalMemory
writeParamMemory
writeSharedMemory
writeTextureMemory
writeLocalMemory
writeRegister

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

Writes content to a hardware register.

Parameters

**dev**
- device index

**sm**
- SM index

**wp**
- warp index

**ln**
- lane index

**regno**
- register index

**val**
- buffer

Returns

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

Since CUDA 3.0.

See also:
writeGenericMemory
writeParamMemory
writeSharedMemory
writeLocalMemory
CUDBGResult (*writeSharedMemory) (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t addr, const void *buf, uint32_t sz)

Writes content to address in the shared memory segment.

Parameters

dev
- device index

sm
- SM index

wp
- warp index

addr
- memory address

buf
- buffer

sz
- size of the buffer

Returns

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

Since CUDA 3.0.

See also:

writeGenericMemory
writeParamMemory
writeLocalMemory
writeRegister
CUDBGResult (*writeUniformPredicates) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t predicates_size, const uint32_t *predicates)

Writes to uniform predicate registers.

Parameters

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **predicates_size**
  - number of predicate registers to write
- **predicates**
  - buffer

Returns

- CUDBG_SUCCESS
- CUDBG_ERROR_INVALID_ARGS
- CUDBG_ERROR_INVALID_DEVICE
- CUDBG_ERROR_INVALID_SM
- CUDBG_ERROR_INVALID_WARP
- CUDBG_ERROR_UNINITIALIZED

Since CUDA 10.0.

See also:

- readUniformPredicate
- writeRegister

CUDBGResult (*writeUniformRegister) (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t regno, uint32_t val)

Writes content to a uniform register.

Parameters

- **dev**
  - device index
- **sm**
  - SM index
- **wp**
  - warp index
- **regno**
  - register number
- **val**
  - value to write
regno
  - register index
val
  - buffer

**Returns**
CUDBG_SUCCESS, CUDBG_ERROR_INVALID_ARGS, CUDBG_ERROR_INVALID_DEVICE,
CUDBG_ERROR_INVALID_SM, CUDBG_ERROR_INVALID_WARP,
CUDBG_ERROR_UNINITIALIZED
Since CUDA 10.0.

**See also:**
writeRegister
readUniformRegisterRange

### 4.2. CUDBGEvent Struct Reference
Event information container.

**CUDBGEvent::cases**
Information for each type of event.

**CUDBGEventKind CUDBGEvent::kind**
Event type.

### 4.3. CUDBGEvent::cases_st Union Reference
struct CUDBGEvent::cases_st::contextCreate_st
CUDBGEvent::cases_st::contextCreate
Information about the context being created.

struct CUDBGEvent::cases_st::contextDestroy_st
CUDBGEvent::cases_st::contextDestroy
Information about the context being destroyed.

struct CUDBGEvent::cases_st::contextPop_st
CUDBGEvent::cases_st::contextPop
Information about the context being popped.

struct CUDBGEvent::cases_st::contextPush_st
CUDBGEvent::cases_st::contextPush
Information about the context being pushed.

struct CUDBGEvent::cases_st::elfImageLoaded_st
CUDBGEvent::cases_st::elfImageLoaded
Information about the loaded ELF image.

struct CUDBGEvent::cases_st::internalError_st
CUDBGEvent::cases_st::internalError
Information about internal errors.

struct CUDBGEvent::cases_st::kernelFinished_st
CUDBGEvent::cases_st::kernelFinished
Information about the kernel that just terminated.

struct CUDBGEvent::cases_st::kernelReady_st
CUDBGEvent::cases_st::kernelReady
Information about the kernel ready to be launched.
4.4. **CUDBGEvent::cases_st::contextCreate_st**

Struct Reference

- **uint64_t**
  - `CUDBGEvent::cases_st::contextCreate_st::context`
  - the context being created.

- **uint32_t**
  - `CUDBGEvent::cases_st::contextCreate_st::dev`
  - device index of the context.

- **uint32_t**
  - `CUDBGEvent::cases_st::contextCreate_st::tid`
  - host thread id (or LWP id) of the thread hosting the context (Linux only).

4.5. **CUDBGEvent::cases_st::contextDestroy_st**

Struct Reference
### 4.6. `CUDBGEvent::cases_st::contextPop_st` Struct Reference

- **uint64_t** `CUDBGEvent::cases_st::contextPop_st::context`  
  the context being popped.

- **uint32_t** `CUDBGEvent::cases_st::contextPop_st::dev`  
  device index of the context.

- **uint32_t** `CUDBGEvent::cases_st::contextPop_st::tid`  
  host thread id (or LWP id) of the thread hosting the context (Linux only).

### 4.7. `CUDBGEvent::cases_st::contextPush_st` Struct Reference
uint64_t CUDBGEvent::cases_st::contextPush_st::context
the context being pushed.

uint32_t CUDBGEvent::cases_st::contextPush_st::dev
device index of the context.

uint32_t CUDBGEvent::cases_st::contextPush_st::tid
host thread id (or LWP id) of the thread hosting the context (Linux only).

4.8. CUDBGEvent::cases_st::elfImageLoaded_st
Struct Reference
context of the kernel.

device index of the kernel.

ELF image handle.

module of the kernel.

ELF image properties.

size of the ELF image (64-bit).

4.9. **CUDBGEvent::cases_st::internalError_st**

Struct Reference
CUDBGResult

CUDBGEvent::cases_st::internalError_st::errorType

Type of the internal error.

4.10. CUDBGEvent::cases_st::kernelFinished_st

Struct Reference
uint64_t
CUDBGEvent::cases_st::kernelFinished_st::context
context of the kernel.

uint32_t
CUDBGEvent::cases_st::kernelFinished_st::dev
device index of the kernel.

uint64_t
CUDBGEvent::cases_st::kernelFinished_st::function
function of the kernel.

uint64_t
CUDBGEvent::cases_st::kernelFinished_st::functionEntry
entry PC of the kernel.

uint64_t
CUDBGEvent::cases_st::kernelFinished_st::gridId
grid index of the kernel.

uint64_t
CUDBGEvent::cases_st::kernelFinished_st::module
module of the kernel.

uint32_t
CUDBGEvent::cases_st::kernelFinished_st::tid
host thread id (or LWP id) of the thread hosting the kernel [Linux only].

4.11. CUDBGEvent::cases_st::kernelReady_st
Struct Reference
**CuDim3**

`CUDBGEvent::cases_st::kernelReady_st::blockDim`

block dimensions of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::context`

context of the kernel.

**uint32_t**

`CUDBGEvent::cases_st::kernelReady_st::dev`

device index of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::function`

function of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::functionEntry`

entry PC of the kernel.

**CuDim3**

`CUDBGEvent::cases_st::kernelReady_st::gridDim`

grid dimensions of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::gridId`

grid index of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::module`

module of the kernel.

**uint64_t**

`CUDBGEvent::cases_st::kernelReady_st::parentGridId`

64-bit grid index of the parent grid.
**uint32_t CUDBGEvent::cases_st::kernelReady_st::tid**

Host thread id (or LWP id) of the thread hosting the kernel [Linux only].

**CUDBGKernelType**

**CUDBGEvent::cases_st::kernelReady_st::type**

The type of the kernel: system or application.

### 4.12. CUDBGEventCallbackData Struct Reference

Event information passed to callback set with setNotifyNewEventCallback function.

**uint32_t CUDBGEventCallbackData::tid**

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

**uint32_t CUDBGEventCallbackData::timeout**

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

### 4.13. CUDBGEventCallbackData40 Struct Reference

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

**uint32_t CUDBGEventCallbackData40::tid**

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

### 4.14. CUDBGGridInfo Struct Reference

Grid info.
CuDim3 CUDBGGridInfo::blockDim
The block dimensions.

uint64_t CUDBGGridInfo::context
The context this grid belongs to.

uint32_t CUDBGGridInfo::dev
The index of the device this grid is running on.

uint64_t CUDBGGridInfo::function
The function corresponding to this grid.

uint64_t CUDBGGridInfo::functionEntry
The entry address of the function corresponding to this grid.

CuDim3 CUDBGGridInfo::gridDim
The grid dimensions.

uint64_t CUDBGGridInfo::gridId64
The 64-bit grid ID of this grid.

uint64_t CUDBGGridInfo::module
The module this grid belongs to.

CUDBGKernelOrigin CUDBGGridInfo::origin
The origin of this grid, CPU or GPU.

uint64_t CUDBGGridInfo::parentGridId
The 64-bit grid ID that launched this grid.

uint32_t CUDBGGridInfo::tid
The host thread ID that launched this grid.

CUDBGKernelType CUDBGGridInfo::type
The type of the grid.
Chapter 5. Data Fields

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

A

- **acknowledgeEvent30**
  - cudbgGetAPI
- **acknowledgeEvents42**
  - cudbgGetAPI
- **acknowledgeSyncEvents**
  - cudbgGetAPI

B

- **blockDim**
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
  - CUDBGGridInfo

C

- **cases**
  - CUDBGEvent
- **clearAttachState**
  - cudbgGetAPI
- **context**
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
  - CUDBGGridInfo
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st
  - CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st
- **contextCreate**
  - CUDBGEvent::CUDBGEvent::cases_st
Data Fields

- getClusterDim
- getDeviceName
- getDevicePCIBusInfo
- getDeviceType
- getElfImage
- getElfImage32
- getElfImageByHandle
- getGridAttribute
- getGridAttributes
- getGridDim
- getGridDim32
- getGridInfo
- getGridInfo55
- getGridStatus
- getGridStatus50
- getHostAddrFromDeviceAddr
- getLoadedFunctionInfo
- getManagedMemoryRegionInfo
- getNextAsyncEvent50
- getNextAsyncEvent55
- getNextEvent
Data Fields

getNextEvent30
cudbgGetAPI
getNextEvent32
cudbgGetAPI
getNextEvent42
cudbgGetAPI
getNextSyncEvent50
cudbgGetAPI
getNextSyncEvent55
cudbgGetAPI
getNumDevices
cudbgGetAPI
getNumLanes
cudbgGetAPI
getNumPredicates
cudbgGetAPI
getNumRegisters
cudbgGetAPI
getNumSMs
cudbgGetAPI
getNumUniformPredicates
cudbgGetAPI
getNumUniformRegisters
cudbgGetAPI
getNumWarps
cudbgGetAPI
getPhysicalRegister30
cudbgGetAPI
getPhysicalRegister40
cudbgGetAPI
getSmType
cudbgGetAPI
getTID
cudbgGetAPI
gridDim
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGGridInfo
gridId
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
gridId64
CUDBGGridInfo
handle

initialize
  cudbgGetAPI
initializeAttachStub
  cudbgGetAPI
internalError
  CUDBGEvent::CUDBGEvent::cases_st
isDeviceCodeAddress
  cudbgGetAPI
isDeviceCodeAddress55
  cudbgGetAPI
kernelFinished
  CUDBGEvent::CUDBGEvent::cases_st
kernelReady
  CUDBGEvent::CUDBGEvent::cases_st
kind
  CUDBGEvent
lookupDeviceCodeSymbol
  cudbgGetAPI
memcheckReadErrorAddress
  cudbgGetAPI
module
  CUDBGGridInfo
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st
origin
  CUDBGGridInfo
P

parentGridId
    CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
    CUDBGGridInfo

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

R

readActiveLanes
    cudbgGetAPI
readBlockIdx
    cudbgGetAPI
readBlockIdx32
    cudbgGetAPI
readBrokenWarps
    cudbgGetAPI
readCallDepth
    cudbgGetAPI
readCallDepth32
    cudbgGetAPI
readCCRegister
    cudbgGetAPI
readClusterIdx
    cudbgGetAPI
readCodeMemory
    cudbgGetAPI
readConstMemory
    cudbgGetAPI
readDeviceExceptionState
    cudbgGetAPI
readDeviceExceptionState80
    cudbgGetAPI
readErrorPC
    cudbgGetAPI
readGenericMemory
    cudbgGetAPI
readGlobalMemory
    cudbgGetAPI
readGlobalMemory31
    cudbgGetAPI
readGlobalMemory55
    cudbgGetAPI
readGridId
cudbgGetAPI
readGridId50
cudbgGetAPI
readLaneException
cudbgGetAPI
readLaneStatus
cudbgGetAPI
readLocalMemory
cudbgGetAPI
readParamMemory
cudbgGetAPI
readPC
cudbgGetAPI
readPinnedMemory
cudbgGetAPI
readPredicates
cudbgGetAPI
readRegister
cudbgGetAPI
readRegisterRange
cudbgGetAPI
readReturnAddress
cudbgGetAPI
readReturnAddress32
cudbgGetAPI
readSharedMemory
cudbgGetAPI
readSyscallCallDepth
cudbgGetAPI
readTextureMemory
cudbgGetAPI
readTextureMemoryBindless
cudbgGetAPI
readThreadIdx
cudbgGetAPI
readUniformPredicates
cudbgGetAPI
readUniformRegisterRange
cudbgGetAPI
readValidLanes
cudbgGetAPI
readValidWarps
    cudbgGetAPI
readVirtualPC
    cudbgGetAPI
readVirtualReturnAddress
    cudbgGetAPI
readVirtualReturnAddress32
    cudbgGetAPI
readWarpState
    cudbgGetAPI
readWarpState60
    cudbgGetAPI
requestCleanupOnDetach
    cudbgGetAPI
requestCleanupOnDetach55
    cudbgGetAPI
resumeDevice
    cudbgGetAPI
resumeWarpsUntilPC
    cudbgGetAPI

S
setBreakpoint
    cudbgGetAPI
setBreakpoint31
    cudbgGetAPI
setKernelLaunchNotificationMode
    cudbgGetAPI
setNotifyNewEventCallback
    cudbgGetAPI
setNotifyNewEventCallback31
    cudbgGetAPI
setNotifyNewEventCallback40
    cudbgGetAPI
singleStepWarp
    cudbgGetAPI
singleStepWarp40
    cudbgGetAPI
singleStepWarp41
    cudbgGetAPI
size
    CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::elfImageLoaded_st
suspendDevice
  cudbgGetAPI

T

tid
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelFinished_st
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPop_st
  CUDBGEventCallbackData
  CUDBGGridInfo
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextCreate_st
  CUDBGEventCallbackData40
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextDestroy_st
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::contextPush_st

timeout
  CUDBGEventCallbackData

type
  CUDBGEvent::CUDBGEvent::cases_st::CUDBGEvent::cases_st::kernelReady_st
  CUDBGGridInfo

U

unsetBreakpoint
  cudbgGetAPI
  unsetBreakpoint31
  cudbgGetAPI

W

writeCCRegister
  cudbgGetAPI
writeGenericMemory
  cudbgGetAPI
writeGlobalMemory
  cudbgGetAPI
writeGlobalMemory31
  cudbgGetAPI
writeGlobalMemory55
  cudbgGetAPI
writeLocalMemory
  cudbgGetAPI
writeParamMemory
  cudbgGetAPI
writePinnedMemory
  cudbgGetAPI
writePredicates  
  cudbgGetAPI
writeRegister  
  cudbgGetAPI
writeSharedMemory  
  cudbgGetAPI
writeUniformPredicates  
  cudbgGetAPI
writeUniformRegister  
  cudbgGetAPI
Chapter 6. Deprecated List

Global CUDBGAPI_st::requestCleanupOnDetach55 (void)
in CUDA 6.0

Class CUDBGEventCallbackData40
in CUDA 4.1.

Global CUDBGAPI_st::singleStepWarp40 (uint32_t dev, uint32_t sm, uint32_t wp)
in CUDA 4.1.

Global CUDBGAPI_st::singleStepWarp41 (uint32_t dev, uint32_t sm, uint32_t wp, uint64_t warpMask)
in CUDA 7.5.

Global CUDBGAPI_st::setBreakpoint31 (uint64_t addr)
in CUDA 3.2.

Global CUDBGAPI_st::unsetBreakpoint31 (uint64_t addr)
in CUDA 3.2.

Global CUDBGAPI_st::readBlockIdx32 (uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *blockIdx)
in CUDA 4.0.
Global CUDBGAPI_st::readCallDepth32 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *depth)
in CUDA 4.0.

Global CUDBGAPI_st::readGlobalMemory31 (uint32_t dev, uint64_t addr, void *buf, uint32_t sz)
in CUDA 3.2.

Global CUDBGAPI_st::readGlobalMemory55 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, void *buf, uint32_t sz)
in CUDA 6.0.

Global CUDBGAPI_st::readGridId50 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t *gridId)
in CUDA 5.5.

Global CUDBGAPI_st::readReturnAddress32 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)
in CUDA 4.0.

Global CUDBGAPI_st::readVirtualReturnAddress32 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t level, uint64_t *ra)
in CUDA 4.0.

Global CUDBGAPI_st::writeGlobalMemory31 (uint32_t dev, uint64_t addr, const void *buf, uint32_t sz)
in CUDA 3.2.

Global CUDBGAPI_st::writeGlobalMemory55 (uint32_t dev, uint32_t sm, uint32_t wp, uint32_t ln, uint64_t addr, const void *buf, uint32_t sz)
in CUDA 6.0.
Global CUDBGAPI_st::getElfImage32
uint32_t dev, uint32_t sm, uint32_t wp, bool relocated, void **elfImage, uint32_t *size)
in CUDA 4.0.

Global CUDBGAPI_st::getGridDim32
uint32_t dev, uint32_t sm, uint32_t wp, CuDim2 *gridDim)
in CUDA 4.0.

Global CUDBGAPI_st::getGridStatus50
uint32_t dev, uint32_t gridId, CUDBGGridStatus *status)
in CUDA 5.5.

Global CUDBGAPI_st::getPhysicalRegister30
uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)
in CUDA 3.1.

Global CUDBGAPI_st::getPhysicalRegister40
uint32_t dev, uint32_t sm, uint32_t wp, uint64_t pc, char *reg, uint32_t *buf, uint32_t sz, uint32_t *numPhysRegs, CUDBGRegClass *regClass)
in CUDA 4.1.

Global CUDBGAPI_st::isDeviceCodeAddress55
uintptr_t addr, bool *isDeviceAddress)
in CUDA 6.0

Global CUDBGNotifyNewEventCallback31
in CUDA 3.2.

Global CUDBGAPI_st::acknowledgeEvent30
(CUDBGEvent30 *event)
in CUDA 3.1.

Global CUDBGAPI_st::acknowledgeEvents42
(void)
in CUDA 5.0.
Global CUDBGAPI_st::getNextAsyncEvent50 (CUDBGEvent50 *event)
in CUDA 5.5.

Global CUDBGAPI_st::getNextEvent30 (CUDBGEvent30 *event)
in CUDA 3.1.

Global CUDBGAPI_st::getNextEvent32 (CUDBGEvent32 *event)
in CUDA 4.0

Global CUDBGAPI_st::getNextEvent42 (CUDBGEvent42 *event)
in CUDA 5.0

Global CUDBGAPI_st::getNextSyncEvent50 (CUDBGEvent50 *event)
in CUDA 5.5.

Global CUDBGAPI_st::setNotifyNewEventCallback31 (CUDBGNotifyNewEventCallback31 callback, void *data)
in CUDA 3.2.

Global CUDBGAPI_st::setNotifyNewEventCallback40 (CUDBGNotifyNewEventCallback40 callback)
in CUDA 4.1.