NAME

Device Management −

Functions

CUresult cuDeviceGet (**CUdevice** *device, int ordinal)
Returns a handle to a compute device.

CUresult cuDeviceGetAttribute (int *pi, **CUdevice_attribute** attrib, **CUdevice** dev)
Returns information about the device.

CUresult cuDeviceGetCount (int *count)
Returns the number of compute-capable devices.

CUresult cuDeviceGetName (char *name, int len, **CUdevice** dev)
Returns an identifer string for the device.

CUresult cuDeviceTotalMem (size_t *bytes, **CUdevice** dev)
Returns the total amount of memory on the device.

Detailed Description

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

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

Function Documentation

CUresult cuDeviceGet (CUdevice * device, int ordinal)

Returns in `*device` a device handle given an ordinal in the range **[0, cuDeviceGetCount()-1].**

Parameters:
device - Returned device handle
ordinal - Device number to get handle for

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

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

See also:
cuDeviceGetAttribute, **cuDeviceGetCount**, **cuDeviceGetName**, **cuDeviceTotalMem**

CUresult cuDeviceGetAttribute (int * pi, CUdevice_attribute attrib, CUdevice dev)

Returns in `*pi` the integer value of the attribute `attrib` on device `dev`. The supported attributes are:

- **CU_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK**: Maximum number of threads per block;
- **CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_X**: Maximum x-dimension of a block;
- **CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Y**: Maximum y-dimension of a block;
- **CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Z**: Maximum z-dimension of a block;
- **CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_X**: Maximum x-dimension of a grid;
- **CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_Y**: Maximum y-dimension of a grid;
- **CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_Z**: Maximum z-dimension of a grid;
- **CU_DEVICE_ATTRIBUTE_MAX_SHARED_MEMORY_PER_BLOCK**: Maximum amount of shared memory available to a thread block in bytes;
- **CU_DEVICE_ATTRIBUTE_TOTAL_CONSTANT_MEMORY**: Memory available on device for __constant__ variables in a CUDA C kernel in bytes;
- **CU_DEVICE_ATTRIBUTE_WARP_SIZE**: Warp size in threads;
- **CU_DEVICE_ATTRIBUTE_MAX_PITCH**: Maximum pitch in bytes allowed by the memory copy functions that involve memory regions allocated through **cuMemAllocPitch()**;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE1D_WIDTH**: Maximum 1D texture width;

- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE1D_LINEAR_WIDTH**: Maximum width for a 1D texture bound to linear memory;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE1D_MIPMAPPED_WIDTH**: Maximum mipmapped 1D texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_WIDTH**: Maximum 2D texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_HEIGHT**: Maximum 2D texture height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LINEAR_WIDTH**: Maximum width for a 2D texture bound to linear memory;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LINEAR_HEIGHT**: Maximum height for a 2D texture bound to linear memory;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LINEAR_PITCH**: Maximum pitch in bytes for a 2D texture bound to linear memory;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_MIPMAPPED_WIDTH**: Maximum mipmapped 2D texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_MIPMAPPED_HEIGHT**: Maximum mipmapped 2D texture height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_WIDTH**: Maximum 3D texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_HEIGHT**: Maximum 3D texture height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_DEPTH**: Maximum 3D texture depth;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_WIDTH_ALTERNATE**: Alternate maximum 3D texture width, 0 if no alternate maximum 3D texture size is supported;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_HEIGHT_ALTERNATE**: Alternate maximum 3D texture height, 0 if no alternate maximum 3D texture size is supported;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE3D_DEPTH_ALTERNATE**: Alternate maximum 3D texture depth, 0 if no alternate maximum 3D texture size is supported;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURECUBEMAP_WIDTH**: Maximum cubemap texture width or height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE1D_LAYERED_WIDTH**: Maximum 1D layered texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE1D_LAYERED_LAYERS**: Maximum layers in a 1D layered texture;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LAYERED_WIDTH**: Maximum 2D layered texture width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LAYERED_HEIGHT**: Maximum 2D layered texture height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURE2D_LAYERED_LAYERS**: Maximum layers in a 2D layered texture;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURECUBEMAP_LAYERED_WIDTH**: Maximum cubemap layered texture width or height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_TEXTURECUBEMAP_LAYERED_LAYERS**: Maximum layers in a cubemap layered texture;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE1D_WIDTH**: Maximum 1D surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE2D_WIDTH**: Maximum 2D surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE2D_HEIGHT**: Maximum 2D surface height;

- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE3D_WIDTH**: Maximum 3D surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE3D_HEIGHT**: Maximum 3D surface height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE3D_DEPTH**: Maximum 3D surface depth;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE1D_LAYERED_WIDTH**: Maximum 1D layered surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE1D_LAYERED_LAYERS**: Maximum layers in a 1D layered surface;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE2D_LAYERED_WIDTH**: Maximum 2D layered surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE2D_LAYERED_HEIGHT**: Maximum 2D layered surface height;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACE2D_LAYERED_LAYERS**: Maximum layers in a 2D layered surface;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACECUBEMAP_WIDTH**: Maximum cubemap surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACECUBEMAP_LAYERED_WIDTH**: Maximum cubemap layered surface width;
- **CU_DEVICE_ATTRIBUTE_MAXIMUM_SURFACECUBEMAP_LAYERED_LAYERS**: Maximum layers in a cubemap layered surface;
- **CU_DEVICE_ATTRIBUTE_MAX_REGISTERS_PER_BLOCK**: Maximum number of 32-bit registers available to a thread block;
- **CU_DEVICE_ATTRIBUTE_CLOCK_RATE**: The typical clock frequency in kilohertz;
- **CU_DEVICE_ATTRIBUTE_TEXTURE_ALIGNMENT**: Alignment requirement; texture base addresses aligned to textureAlign bytes do not need an offset applied to texture fetches;
- **CU_DEVICE_ATTRIBUTE_TEXTURE_PITCH_ALIGNMENT**: Pitch alignment requirement for 2D texture references bound to pitched memory;
- **CU_DEVICE_ATTRIBUTE_GPU_OVERLAP**: 1 if the device can concurrently copy memory between host and device while executing a kernel, or 0 if not;
- **CU_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT**: Number of multiprocessors on the device;
- **CU_DEVICE_ATTRIBUTE_KERNEL_EXEC_TIMEOUT**: 1 if there is a run time limit for kernels executed on the device, or 0 if not;
- **CU_DEVICE_ATTRIBUTE_INTEGRATED**: 1 if the device is integrated with the memory subsystem, or 0 if not;
- **CU_DEVICE_ATTRIBUTE_CAN_MAP_HOST_MEMORY**: 1 if the device can map host memory into the CUDA address space, or 0 if not;
- **CU_DEVICE_ATTRIBUTE_COMPUTE_MODE**: Compute mode that device is currently in. Available modes are as follows:
  - **CU_COMPUTEMODE_DEFAULT**: Default mode - Device is not restricted and can have multiple CUDA contexts present at a single time.
  - **CU_COMPUTEMODE_EXCLUSIVE**: Compute-exclusive mode - Device can have only one CUDA context present on it at a time.
  - **CU_COMPUTEMODE_PROHIBITED**: Compute-prohibited mode - Device is prohibited from creating new CUDA contexts.
  - **CU_COMPUTEMODE_EXCLUSIVE_PROCESS**: Compute-exclusive-process mode - Device can have only one context used by a single process at a time.
- **CU_DEVICE_ATTRIBUTE_CONCURRENT_KERNELS**: 1 if the device supports executing multiple kernels within the same context simultaneously, or 0 if not. It is not guaranteed that multiple

kernels will be resident on the device concurrently so this feature should not be relied upon for correctness;

- **CU_DEVICE_ATTRIBUTE_ECC_ENABLED**: 1 if error correction is enabled on the device, 0 if error correction is disabled or not supported by the device;
- **CU_DEVICE_ATTRIBUTE_PCI_BUS_ID**: PCI bus identifier of the device;
- **CU_DEVICE_ATTRIBUTE_PCI_DEVICE_ID**: PCI device (also known as slot) identifier of the device;
- **CU_DEVICE_ATTRIBUTE_TCC_DRIVER**: 1 if the device is using a TCC driver. TCC is only available on Tesla hardware running Windows Vista or later;
- **CU_DEVICE_ATTRIBUTE_MEMORY_CLOCK_RATE**: Peak memory clock frequency in kilohertz;
- **CU_DEVICE_ATTRIBUTE_GLOBAL_MEMORY_BUS_WIDTH**: Global memory bus width in bits;
- **CU_DEVICE_ATTRIBUTE_L2_CACHE_SIZE**: Size of L2 cache in bytes. 0 if the device doesn't have L2 cache;
- **CU_DEVICE_ATTRIBUTE_MAX_THREADS_PER_MULTIPROCESSOR**: Maximum resident threads per multiprocessor;
- **CU_DEVICE_ATTRIBUTE_UNIFIED_ADDRESSING**: 1 if the device shares a unified address space with the host, or 0 if not;
- **CU_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY_MAJOR**: Major compute capability version number;
- **CU_DEVICE_ATTRIBUTE_COMPUTE_CAPABILITY_MINOR**: Minor compute capability version number;
- **CU_DEVICE_ATTRIBUTE_GLOBAL_L1_CACHE_SUPPORTED**: 1 if device supports caching globals in L1 cache, 0 if caching globals in L1 cache is not supported by the device;
- **CU_DEVICE_ATTRIBUTE_LOCAL_L1_CACHE_SUPPORTED**: 1 if device supports caching locals in L1 cache, 0 if caching locals in L1 cache is not supported by the device;
- **CU_DEVICE_ATTRIBUTE_MAX_SHARED_MEMORY_PER_MULTIPROCESSOR**: Maximum amount of shared memory available to a multiprocessor in bytes; this amount is shared by all thread blocks simultaneously resident on a multiprocessor;
- **CU_DEVICE_ATTRIBUTE_MAX_REGISTERS_PER_MULTIPROCESSOR**: Maximum number of 32-bit registers available to a multiprocessor; this number is shared by all thread blocks simultaneously resident on a multiprocessor;
- **CU_DEVICE_ATTRIBUTE_MANAGED_MEMORY**: 1 if device supports allocating managed memory on this system, 0 if allocating managed memory is not supported by the device on this system.
- **CU_DEVICE_ATTRIBUTE_MULTI_GPU_BOARD**: 1 if device is on a multi-GPU board, 0 if not.
- **CU_DEVICE_ATTRIBUTE_MULTI_GPU_BOARD_GROUP_ID**: Unique identifier for a group of devices associated with the same board. Devices on the same multi-GPU board will share the same identifier.

Parameters:

pi - Returned device attribute value
attrib - Device attribute to query
dev - Device handle

Returns:

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

Note:

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

See also:
cuDeviceGetCount, **cuDeviceGetName**, **cuDeviceGet**, **cuDeviceTotalMem**

CUresult cuDeviceGetCount (int * count)
Returns in `*count` the number of devices with compute capability greater than or equal to 1.0 that are available for execution. If there is no such device, **cuDeviceGetCount()** returns 0.

Parameters:
count - Returned number of compute-capable devices

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

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

See also:
cuDeviceGetAttribute, **cuDeviceGetName**, **cuDeviceGet**, **cuDeviceTotalMem**

CUresult cuDeviceGetName (char * name, int len, CUdevice dev)
Returns an ASCII string identifying the device `dev` in the NULL-terminated string pointed to by `name`. `len` specifies the maximum length of the string that may be returned.

Parameters:
name - Returned identifier string for the device
len - Maximum length of string to store in `name`
dev - Device to get identifier string for

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

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

See also:
cuDeviceGetAttribute, **cuDeviceGetCount**, **cuDeviceGet**, **cuDeviceTotalMem**

CUresult cuDeviceTotalMem (size_t * bytes, CUdevice dev)
Returns in `*bytes` the total amount of memory available on the device `dev` in bytes.

Parameters:
bytes - Returned memory available on device in bytes
dev - Device handle

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

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

See also:
cuDeviceGetAttribute, **cuDeviceGetCount**, **cuDeviceGetName**, **cuDeviceGet**,

Author
Generated automatically by Doxygen from the source code.