

---

---

NVIDIA CUDA  
Windows XP and Vista Release Notes  
Version 2.0

---

---

---

## New Features

---

### Hardware Support

- o Additional hardware support:
  - GeForce GTX 280
  - GeForce GTX 260
  - GeForce 9800 GX2
  - GeForce 9800 GTX
  - GeForce 9600 GT
  - GeForce 8800 GS
  - GeForce 8600 GTS
  - Quadro FX 3700
  - Quadro NVS 130M
  - Quadro NVS 135M
  - Quadro NVS 140M
  - Quadro NVS 140M
  - Quadro NVS 135M
  - Quadro NVS 130M
  - Quadro FX 3600M

### Platform Support

- o Additional OS support
  - Windows Vista 32-bit
  - Windows Vista 64-bit

### New ISA Support SM 1.2

- o Support for any() and all() intrinsics
- o Support for atomic operations on shared memory
- o Support for 64-bit atomic operations on global memory
  - atomicAdd() (unsigned 64-bit int)
  - atomicExch() (unsigned 64-bit int)
  - atomicCAS() (unsigned 64-bit int)

### Double Precision Computing with SM 1.3

- o Compiler support for double-precision math
  - Datatype 'double' compiles to native FP64 types when using SM 1.3
- Note that math functions in the CUDA math library are overloaded. In general, there are three prototypes for each math function:
  - (1) double <func-name>(double), e.g. double log(double)
  - (2) float <func-name>(float), e.g. float log(float)
  - (3) float <func-name>f(float), e.g. float logf(float)
- In particular, note that passing a float argument always results in a float result [variants (2) and (3) above].
- o CUBLAS Library Support
  - Added the BLAS1 functions:
    - \* cublasIdamax()

- \* cublasIdamin()
- \* cublasDasum()
- \* cublasDaxpy()
- \* cublasDcopy()
- \* cublasDdot()
- \* cublasDnrm2()
- \* cublasDrot()
- \* cublasDrotg()
- \* cublasDrotm()
- \* cublasDrotmg()
- \* cublasDscal()
- \* cublasDswap()
- Added the BLAS2 functions:
  - \* cublasDgemv()
  - \* cublasDger()
  - \* cublasDsyr()
  - \* cublasDtrsv()
- Added the BLAS3 functions:
  - \* cublasDgemm()
  - \* cublasDsymm()
  - \* cublasDsyrk()
  - \* cublasDsyr2k()
  - \* cublasDtrmm()
  - \* cublasDtrsm()
  - \* cublasZgemm()

#### API Features

- o 3D texture API
  - cudaMalloc3D
  - cudaMalloc3DArray
  - cudaMemset3D
  - cu(da)Memcpy3D
  - cu(da)Memcpy3DAsync
  - cuArray3DCreate
  - cuArray3DGetDescriptor
  - CUDA\_MEMCPY3D and CUDA\_ARRAY3D\_DESCRIPTOR structures
- o Improved Direct3D interoperability API
  - cudaD3D9SetDirect3DDevice
  - cu(da)D3D9GetDirect3DDevice
  - cu(da)D3D9RegisterResource
  - cu(da)D3D9UnregisterResource
  - cu(da)D3D9MapResources
  - cu(da)D3D9UnmapResources
  - cu(da)D3D9ResourceSetMapFlags
  - cu(da)D3D9ResourceGetMappedPointer
  - cu(da)D3D9ResourceGetMappedSize
  - cu(da)D3D9ResourceGetMappedPitch
  - cu(da)D3D9ResourceGetSurfaceDimensions
  - cuD3D9CtxCreate
- o Improved OpenGL interoperability API
  - cuGLCtxCreate
- o Context migration API
  - cuCtxAttach
  - cuCtxDestroy
  - cuCtxDetach
  - cuCtxSynchronize
  - cuCtxPushCurrent

- cuCtxPopCurrent
- o Async constant memory update
  - cudaMemcpyToSymbolAsync
- o Improved device attribute query
  - cuDeviceGetAttribute

#### Performance Enhancements

- o Improved device->array memcpy performance

---

#### Known Issues

---

##### Vista Specific Issues:

- o The Windows desktop must be extended onto the GPU in order to run CUDA.
- o The TESLA C870 and D870 are not supported on Windows Vista.
- o Individual kernels are limited to a 2-second runtime by Windows Vista. Kernels that run for longer than 2 seconds will trigger the Timeout Detection and Recovery (TDR) mechanism. For more information, see [http://www.microsoft.com/whdc/device/display/wddm\\_timeout.mspx](http://www.microsoft.com/whdc/device/display/wddm_timeout.mspx).
- o The CUDA Profiler does not support performance counter events on Windows Vista. All profiler configuration regarding performance counter events is ignored.
- o On Windows Vista, asynchronous memory copies do not support GPU overlap. CU\_DEVICE\_ATTRIBUTE\_GPU\_OVERLAP will be 0 for all devices.

##### XP Specific Issues:

- o Individual GPU program launches are limited to a run time of less than 5 seconds on a GPU with a display attached. Exceeding this time limit usually causes a launch failure reported through the CUDA driver or the CUDA runtime. GPUs without a display attached are not subject to the 5 second runtime restriction. For this reason it is recommended that CUDA be run on a GPU that is NOT attached to a display and does not have the Windows desktop extended onto it. In this case, the system must contain at least one NVIDIA GPU that serves as the primary graphics adapter.

##### Issues Common to XP and Vista:

- o Applications that try to use too much memory may cause a CUDA memcpy or kernel to fail with the error CUDA\_ERROR\_OUT\_OF\_MEMORY. If this happens, the CUDA Context is placed into an error state and must be destroyed and recreated if the application wants to continue using CUDA.
- o Malloc may fail due to running out of virtual memory space.

The address space limitation is fixed by a Microsoft issued hotfix. Please install the patch located at <http://support.microsoft.com/kb/940105> if this is an issue. Windows Vista SP1 includes this hotfix.

- o When two GPUs are run in SLI mode, only one of the GPUs will be available to the user for executing CUDA programs.
- o Inline assembly is not supported in host code, resulting in the error message "error: a microsoft style "asm" declaration is not allowed here".
- o The intermediate source code that nvcc generates can trigger an internal error when compiled with Microsoft Studio Visual C++ 7.1. The error message you may receive will look similar to the following:

```
yourfile.cu : fatal error C1001: INTERNAL COMPILER ERROR
(compiler file 'mscl.cpp', line 2701)
Please choose the Technical Support command on the Visual C++
Help menu, or open the Technical Support help file for more information
```

This is a known problem in the Microsoft Studio Visual C++ 7.1 compiler. More information on this internal error can be found on Microsoft's support website: <http://support.microsoft.com/kb/824389/en-gb>

- o When using Microsoft Studio Visual 8.0, it is required that Service Pack 1 be installed. Certain Windows C++ header files will cause a crash in cudafe without it.
- o The default compilation mode for host code is now C++. To restore the old behavior, use the option `--host-compilation=c`

---

## Open64 Sources

---

The Open64 source files are controlled under terms of the GPL license. Current and previously released versions are located via anonymous ftp at [download.nvidia.com](http://download.nvidia.com) in the CUDAOpen64 directory.

---

## Revision History

---

06/2008 - Version 2.0  
05/2008 - Version 2.0 Beta  
11/2007 - Version 1.1  
06/2007 - Version 1.0  
06/2007 - Version 0.9  
02/2007 - Version 0.8 - Initial public Beta

---

## More Information

---

For more information and help with CUDA, please visit  
<http://www.nvidia.com/cuda>