
NVIDIA CUDA
Windows Release Notes
Version 1.1

IMPORTANT NOTE: Display Driver Integration and CUDA 1.0 Compatibility

With CUDA v1.1, the driver API functions have been integrated with the display driver. The CUDA driver (nvcuda.dll) is installed in the %SystemRoot%\system32 directory along with display driver files such as nv4_disp.dll (the display driver) and nvoglnt.dll (the OpenGL driver). As a result, driver API applications must be rebuilt using CUDA 1.1 in order to work.

In addition, CUDA 1.0 driver API applications *are not* supported by CUDA 1.1-compatible display drivers. The CUDA 1.0 driver (cuda.dll) does not work properly on systems running CUDA 1.1 display drivers. If a CUDA 1.0 application is run on a CUDA 1.1 display driver, it will not be able to create a CUDA context.

New Features

Hardware Support

o Additional hardware support:

- Tesla C870
- Tesla D870
- Tesla S870

- Quadro FX 1700
- Quadro FX 570
- Quadro FX 370
- Quadro NVS 130M
- Quadro NVS 135M
- Quadro NVS 140M
- Quadro NVS 290
- Quadro NVS 320M
- Quadro FX 1600M
- Quadro FX 570M
- Quadro FX 360M
- Quadro Plex 1000 Model IV
- Quadro Plex 1000 Model S4

- GeForce 8800 GT
- GeForce 8400 GS
- GeForce 8800M GTX
- GeForce 8800M GTS
- GeForce 8700M GT
- GeForce 8600M GT
- GeForce 8600M GS
- GeForce 8400M GT
- GeForce 8400M GS
- GeForce 8400M G

Compiler and Toolchain

- o Microsoft Visual C++ 6.0 is no longer supported.
- o `__global__` functions are now always mangled. When using the driver API developers must declare these with `'extern "C"'`.
- o device emulation mode no longer requires cuda.dll.
- o C++ language support
 - host code only, alpha quality
 - developers must "opt in" with the `--host-compilation=C++` option
- o Added device functions:
 - `__clz`
 - `__clzll`
 - `__ffs`

- __ffsll
- __sad
- __float2ll_rz
- __float2ll_rn
- __float2ull_rz
- __ll2float_rn
- __ull2float_rn
- __usad
- o Removed the float-valued atomicCAS device function

Mathematical Functions

- o Added math functions:
 - llrintf()
 - llroundf()
- o Improved accuracy of the functions:
 - cosf
 - powf
 - sinf
 - sincosf
 - tanf

Asynchronous Memcpy Support

- o Streams and Asynchronous memcpy functions
 - CPU/GPU concurrency and compute/host<->device memcpy concurrency
- o Events (enable high-precision timing and CPU/GPU synchronization)

Miscellaneous

- o New cuDeviceGetAttribute function obsoletes cuDeviceGetProperties
 - Also enables applications to query whether the hardware can process and copy data concurrently
- o cuD3D9GetDevice function enables Direct3D interoperability on multi-GPU hardware configurations

CUFFT Library

- o Performance improvements

CUBLAS Library

- o Performance improvements
- o Added the functions:
 - cublasCrot()
 - cublasCrotg()
 - cublasCsrot()

Major Bug Fixes

- o OpenGL interoperability now works on multi-GPU systems

Known Issues

- o During compilation, the compiler may emit up to four temporary filenames to stdout. This is a benign side-effect of a change in the preprocessor for the Windows version of the CUDA compiler.

If the `-keep` option is specified to `nvcc.exe`, the filenames are derived from the source file being compiled, e.g.:

```
foo.cudaf1.gpu
foo.cudafe2.gpu
foo.cudaf1.c
foo.cu.c
```

If the `-keep` option is not specified to `nvcc.exe`, the filenames take the form `"tmpxft_*.gpu,"` e.g.

```
tmpxft_00001278_00000000-3.gpu
```

- 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 run time 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.

- o When two 8800GTX GPUs are run in SLI mode, `cudaGetDeviceCount()` correctly reports a single device, but `cudaDeviceProperties()` only returns the amount of memory on the first card, instead of the total amount of memory installed on both cards used for SLI.
- o The compiler does not emit error messages in a format that can be processed by the Microsoft Visual Studio environment.
- 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 'msc1.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 On systems with multiple GPUs installed or systems with a single GPU and DualView enabled, OpenGL interoperability always copies shared buffers through host memory.
- o Current hardware limits the number of asynchronous memcopies that can be overlapped with kernel execution. Overlap is also limited to kernels executing for less than 1 second. These limitations are expected to improve on future hardware.
- o In order to reduce unwanted CPU utilization, the following APIs have been modified to yield the CPU when the device is busy.
 - `cuCtxSynchronize`
 - `cuEventSynchronize`
 - `cuStreamSynchronize`
 - `cudaThreadSynchronize`
 - `cudaEventSynchronize`
 - `cudaStreamSynchronize`
- o When the profiler gathers performance signals on G80-based products, the driver reduces the clock rate on the device. If the CUDA app crashes or otherwise exits uncleanly, the clocks will not be reset to their previous values. The system must be rebooted to restore the original clock rate.

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 in the `CUDAOpen64` directory.

Revision History

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>