NVIDIA CUDA GETTING STARTED GUIDE FOR MICROSOFT WINDOWS

DU-05349-001_v04 | April 2012

Installation and Verification on Windows

DOCUMENT CHANGE HISTORY

DU-05349-001_v04

Version	Date	Authors	Description of Change
01	April 20, 2010	CW, TS	Release
02	August 19, 2010	CW	Updated for CUDA Toolkit 3.2
03	March 3, 2011	CW	Updated for CUDA Toolkit 4.0
04	April 6, 2012	CW, RC	Added information on Visual Studio 2010 build customizations

TABLE OF CONTENTS

INTRODUCTION

NVIDIA® CUDA™ is a general purpose parallel computing architecture introduced by NVIDIA. It includes the CUDA Instruction Set Architecture (ISA) and the parallel compute engine in the GPU. To program to the CUDA architecture, developers can use C, one of the most widely used high-level programming languages, which can then be run at great performance on a CUDA-enabled processor.

The CUDA architecture and its associated software were developed with several design goals in mind:

- Provide a small set of extensions to standard programming languages, like C, that enable a straightforward implementation of parallel algorithms. With CUDA C, programmers can focus on the task of parallelization of the algorithms rather than spending time on their implementation.
- Support heterogeneous computation where applications use both the CPU and GPU. Serial portions of applications are run on the CPU, and parallel portions are offloaded to the GPU. As such, CUDA can be incrementally applied to existing applications. The CPU and GPU are treated as separate devices that have their own memory spaces. This configuration also allows simultaneous computation on both the CPU and GPU without contention for memory resources.

CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

This guide will show you how to install and check the correct operation of the CUDA development tools.

SYSTEM REQUIREMENTS

To use CUDA on your system, you will need the following installed:

- CUDA-enabled GPU
- Microsoft Windows XP, Vista, or 7 or Windows Server 2003 or 2008.
- Device driver
- CUDA software (available at no cost from http://www.nvidia.com/cuda)
- Microsoft Visual Studio 2005, 2008, or 2010, or the corresponding versions of Microsoft Visual C++ Express

ABOUT THIS DOCUMENT

This document is intended for readers familiar with Microsoft Windows XP, Microsoft Windows Vista, or Microsoft Windows 7 operating systems and the Microsoft Visual Studio environment. You do not need previous experience with CUDA or experience with parallel computation.

INSTALLING CUDA DEVELOPMENT TOOLS

The installation of CUDA development tools on a system running the appropriate version of Windows consists of four simple steps:

- Verify the system has a CUDA-enabled GPU
- Download the CUDA software
- Install the driver for Windows XP, Windows Vista or Windows 7 (if necessary)
- Install the CUDA software

Test your installation by compiling and running one of the sample programs in the CUDA software to validate that the hardware and software are running correctly and communicating with each other.

VERIFY YOU HAVE A CUDA-ENABLED SYSTEM

Many NVIDIA products today contain CUDA-enabled GPUs. These include:

- NVIDIA GeForce® 8, 9, 200, 400, 500, and 600 series GPUs
- NVIDIA Tesla™ computing solutions
- Many of the NVIDIA Quadro® products

An up-to-date list of CUDA-enabled GPUs can be found on the NVIDIA CUDA Web site at http://www.nvidia.com/object/cuda_gpus.html.

The Release Notes for the CUDA Toolkit also contain a list of supported products.

To verify which video adapter your Windows system uses, open the Control Panel (**Start→Control Panel)** and double click on **System**. In the **System Properties** window that opens, click the **Hardware** tab, then **Device Manager**. Expand the **Display adapters** entry. There you will find the vendor name and model of your graphics card.

DOWNLOAD THE CUDA SOFTWARE

The CUDA software is available at http://www.nvidia.com/object/cuda_get.html.

Choose the platform you are using and download the following:

- The CUDA Driver
  The CUDA Driver is integrated into the NVIDIA ForceWare® graphics driver, which can be downloaded from http://www.nvidia.com/drivers. To use the CUDA Toolkit, you must have at least the version of the NVIDIA graphics driver specified in the *CUDA Toolkit Release Notes*. A developer version of the NVIDIA graphics driver is available from the CUDA software download page that corresponds to the version of the CUDA Toolkit also available for download there, but any newer NVIDIA driver will work as well.
- The CUDA Toolkit
  The CUDA Toolkit contains the tools needed to compile and build a CUDA application in conjunction with Microsoft Visual Studio. It includes tools, libraries, header files, and other resources.
- The GPU Computing SDK
  The GPU Computing SDK (software development kit) includes sample projects that have all the necessary project configuration and build files to perform one-click builds using Microsoft Visual Studio.

These software packages are available for both 32-bit Windows and 64-bit Windows.

Before installing these packages, you should read the *Release Notes* bundled with each, as these notes provide details on installation and software functionality. To identify the version of your NVIDIA driver, open the NVIDIA Control Panel by right clicking on the desktop and selecting **NVIDIA Control Panel**. Click the **System Information** button in the lower left corner of the main panel to display a dialog box that specifies the version of the driver installed on your system.

Note: New versions of the CUDA Toolkit typically require new versions of the NVIDIA driver as well, so always verify that you are running the correct release of the driver for the version of the CUDA Toolkit that you are using.

INSTALL THE CUDA SOFTWARE

Use the following procedure to install the CUDA software:

1. **Install the CUDA Toolkit.**
   Install the CUDA Toolkit by executing the Toolkit installer package and following the on-screen prompts. The CUDA Toolkit installation defaults to **`C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v#.#`**, where #.# is version number 3.2 or higher. This directory contains the following:

   - **`Bin\`** - the compiler executables and runtime libraries
   - **`Include\`** - the header files needed to compile CUDA programs
   - **`Lib\`** - the library files needed to link CUDA programs
   - **`Doc\`** - the *CUDA C Programming Guide*, *CUDA C Best Practices Guide*, documentation for the CUDA libraries, and other CUDA Toolkit-related documentation

   Note: CUDA Toolkit versions 3.1 and earlier installed into **`C:\CUDA`** by default, requiring prior CUDA Toolkit versions to be uninstalled before the installation of new versions. Beginning with CUDA Toolkit 3.2, multiple CUDA Toolkit versions can be installed simultaneously.

2. **Install the GPU Computing SDK.**
   Install the GPU Computing SDK by executing the installer package and following the on-screen prompts. The GPU Computing SDK is installed in **`C:\Documents and Settings\All Users\Application Data\NVIDIA Corporation\NVIDIA GPU Computing SDK`** (or under **`%ProgramData%\NVIDIA Corporation\...`** on Windows Vista or later) and contains source code for many example problems and templates for Microsoft Visual Studio.

VERIFY THE INSTALLATION

Before continuing, it is important to verify that the CUDA programs can find and communicate correctly with the CUDA-enabled hardware. To do this, you need to compile and run some of the included sample programs.

Compiling the Examples

The version of the CUDA Toolkit can be checked by running **nvcc -V** in a Command Prompt window. You can display a Command Prompt window by going to:
Start→All Programs→Accessories→Command Prompt

The GPU Computing SDK includes sample programs in both *source* and *compiled* form. To verify a correct configuration of the hardware and software, it is highly recommended that you run the **bandwidthTest** program located in **C:\Documents and Settings\All Users\Application Data\NVIDIA Corporation\NVIDIA GPU Computing SDK\C\bin\win32\Release**, presuming that you used the default installation directory structure. (On 64-bit versions of Windows, the directory name ends with **\win64\Release**.) If CUDA is installed and configured correctly, the output should look similar to Figure 1.

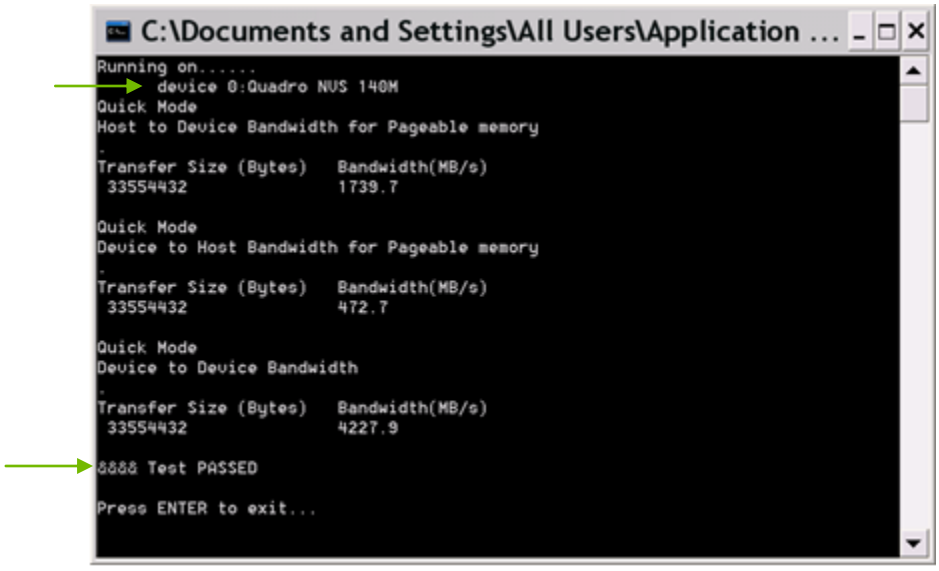

Figure 1. Valid Results from Sample CUDA bandwidthTest Program

The device name (second line) and the bandwidth numbers vary from system to system. The important items are the second line, which confirms a CUDA device was found, and the second-to-last line, which confirms that all necessary tests passed.

If the tests do not pass, make sure you do have a CUDA-enabled NVIDIA GPU on your system and make sure it is properly installed.

To see a graphical representation of what CUDA can do, run the sample Particles executable in:

- For Windows XP:
  `c:\Documents and Settings\All Users\Application Data\NVIDIA GPU Computing SDK\C\bin\win32\Release` (or …**`\win64\Release`** on 64-bit Windows)
- For Windows Vista and Windows 7:
  `C:\ProgramData\NVIDIA Corporation\NVIDIA GPU Computing SDK\C\bin\win32\Release` (or …**`\win64\Release`** on 64-bit Windows)

COMPILING CUDA PROGRAMS

The project files in the GPU Computing SDK have been designed to provide simple, one-click builds of the programs that include all source code. To build the 32-bit or 64-bit Windows projects (for release or debug mode), use the provided **`*.sln`** solution files for Microsoft Visual Studio 2005, 2008, or 2010 (and likewise for the corresponding versions of Microsoft Visual C++ Express Edition). You can use either the solution files located in each of the examples directories in **`NVIDIA GPU Computing SDK\C\src`** or the global solution files **`Release*.sln`** located in **`NVIDIA GPU Computing SDK\C\src`**.

COMPILING SAMPLE PROJECTS

The **`bandwidthTest`** project is a good sample project to build and run. It is located in the **`…NVIDIA Corporation\NVIDIA GPU Computing SDK\C\src\bandwidthTest`** directory.

The output is placed in **`NVIDIA GPU Computing SDK\C\bin\win32\Debug`**. (As mentioned previously, the **`\win32`** segment of this address will be **`\win64`** on 64-bit versions of Windows.) This location presumes that you used the default installation directory structure.

Build the program using the appropriate solution file and run the executable. If all works correctly, the output should be similar to Figure 1.

SAMPLE PROJECTS

The sample projects come in two configurations: debug and release (where release contains no debugging information).

A few of the example projects require some additional setup. The **simpleD3D9** example requires the system to have a Direct3D SDK installed and the Visual C++ directory paths (located in **Tools→Options**...) properly configured. Consult the Direct3D documentation for additional details.

Most samples link to a utility library called **cutil** , the source code for which is in **NVIDIA GPU Computing SDK\C\common\src**. The release versions of these samples link to **cutil32.lib** (or **cutil64.lib**) and dynamically load **cutil32.dll** (or **cutil64.dll**). The debug versions of these samples link to **cutil32D.lib** and dynamically load **cutil32D.dll** (or their 64-bit equivalents on 64-bit versions of Windows).

To build the Win32 release and/or debug configurations of the **cutil library**, use the solution files located in **NVIDIA GPU Computing SDK\C\common**. The output of the compilation process should be placed in **NVIDIA GPU Computing SDK\C\common\lib**:

cutil32.lib and **cutil32D.lib** (or **cutil64.lib** and **cutil64D.lib**) are the release and debug import libraries.

cutil32.dll and **cutil32D.dll** (or **cutil64.dll** and **cutil64D.dll**) are the release and debug dynamic-link libraries, which also are copied to **NVIDIA GPU Computing SDK\C\bin\win32\release** and **NVIDIA GPU Computing SDK\C\bin\win32\debug** respectively. (Substitute **\win64** for **\win32** on 64-bit Windows.)

Note: The **cutil** library is primarily intended as a tool for streamlining the GPU Computing SDK samples; its use outside of the SDK samples is not recommended, as the API for the library may change at any time.

These sample projects also make use of the **$CUDA_PATH** environment variable to locate the CUDA Toolkit and a **.rules** file to locate and configure the **nvcc** compiler. The environment variable is set automatically and the **.rules** file is installed automatically as part of the CUDA Toolkit installation process. The **.rules** file is installed into **$VisualStudioInstallDir\VC\VCProjectDefaults**. You can reference this **.rules** file from your Visual Studio project files when building your own CUDA applications.

BUILD CUSTOMIZATIONS FOR EXISTING PROJECTS

When adding CUDA acceleration to existing applications, the relevant Visual Studio project file must be updated to include CUDA build customizations. For Visual Studio 2010, this can done using one of the following two methods:

1. Open the Visual Studio 2010 project, right click on the project name, and select "Build Customizations...", then select the CUDA Toolkit version you would like to target.
2. Alternatively, you can configure your project always to build with the most recently installed version of the CUDA Toolkit. First add a CUDA build customization to your project as above. Then, right click on the project name and select "Properties". Under "CUDA C/C++", select "Common", and set the "CUDA Toolkit Custom Dir" field to "$(CUDA_PATH)" (without the quotes).

While Option 2 will allow your project to automatically use any new CUDA Toolkit version you may install in the future, selecting the toolkit version explicitly as in Option 1 is often better in practice, because if there are new CUDA configuration options added to the build customization rules accompanying the newer toolkit, you would not see those new options using Option 2.

Note for advanced users: if you wish to try building your project against a newer CUDA Toolkit without making changes to any of your project files, go to the Visual Studio 2010 command prompt, change the current directory to the location of your project, and execute a command such as the following:

```
msbuild <projectname.extension> /t:Rebuild
 /p:CudaToolkitDir="drive:/path/to/new/toolkit/"
```

ADDITIONAL CONSIDERATIONS

Now that you have CUDA-enabled hardware and the software installed, you can examine and enjoy the numerous included programs. To begin using CUDA to accelerate the performance of your own applications, consult the *CUDA C Programming Guide*, located in the CUDA Toolkit documentation directory.

For technical support on programming questions, consult and participate in the bulletin board and mailing list at http://forums.nvidia.com/index.php?showforum=71.

Notice

Trademarks

Copyright