NVIDIA CUDA GETTING STARTED GUIDE FOR MICROSOFT WINDOWS

DU-05349-001_v6.5 | August 2014

Installation and Verification on Windows

TABLE OF CONTENTS

Chapter 1.
INTRODUCTION

CUDA® is a parallel computing platform and programming model invented by NVIDIA. It enables dramatic increases in computing performance by harnessing the power of the graphics processing unit (GPU).

CUDA was 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/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 the CPU and GPU without contention for memory resources.

CUDA-capable GPUs have hundreds of cores that can collectively run thousands of computing threads. These cores have shared resources including a register file and a shared 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.

1.1. System Requirements

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

▸ A CUDA-capable GPU
▸ A supported version of Microsoft Windows
▸ A supported version of Microsoft Visual Studio
▸ the NVIDIA CUDA Toolkit (available at http://developer.nvidia.com/cuda-downloads)

The next two tables list the currently supported Windows operating systems and compilers.

Table 1   Windows Operating System Support in CUDA 6.5

Operating System	Native x86_64	Native x86_32(*)	Cross (x86_32 on x86_64)
Windows 8.1	YES	DEPRECATED	YES
Windows 7	YES	DEPRECATED	YES
Windows XP	NO	DEPRECATED	NO
Windows Server 2012 R2	YES	—	YES
Windows Server 2008 R2	YES	DEPRECATED	YES

Table 2   Windows Compiler Support in CUDA 6.5

Compiler	IDE	Native x86_64	Native x86_32	Cross (x86_32 on x86_64)
Visual C++ 12.0	Visual Studio 2013	YES	DEPRECATED	YES
	Visual Studio Express 2013 for Windows Desktop	—	DEPRECATED	YES
	Visual Studio Express 2013 for Windows	—	—	—
Visual C++ 11.0	Visual Studio 2012	YES	DEPRECATED	YES
Visual C++ 10.0	Visual Studio 2010	YES	DEPRECATED	YES

(*) Only native development using the CUDA Toolkit on x86_32 is deprecated. Deployment and execution of CUDA applications using the CUDA Driver on x86_32 is still supported. To create 32-bit CUDA applications, use the cross-development capabilities of the CUDA Toolkit on x86_64.

1.2. About This Document

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

Chapter 2.
INSTALLING CUDA DEVELOPMENT TOOLS

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

▸ Verify the system has a CUDA-capable GPU.
▸ Download the NVIDIA CUDA Toolkit.
▸ Install the NVIDIA CUDA Toolkit.
▸ Test that the installed software runs correctly and communicates with the hardware.

2.1. Verify You Have a CUDA-Capable GPU

To verify that your GPU is CUDA-capable, 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. If it is an NVIDIA card that is listed in http://developer.nvidia.com/cuda-gpus, your GPU is CUDA-capable.

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

2.2. Download the NVIDIA CUDA Toolkit

The NVIDIA CUDA Toolkit is available at http://developer.nvidia.com/cuda-downloads.

Choose the platform you are using and download the NVIDIA CUDA Toolkit

The CUDA Toolkit contains the CUDA driver and tools needed to create, build and run a CUDA application as well as libraries, header files, CUDA samples source code, and other resources.

Download Verification

The download can be verified by comparing the MD5 checksum posted at http://developer.nvidia.com/cuda-downloads/checksums with that of the downloaded

file. If either of the checksums differ, the downloaded file is corrupt and needs to be downloaded again.

To calculate the MD5 checksum of the downloaded file, follow the instructions at http://support.microsoft.com/kb/889768.

2.3. Install the CUDA Software

Before installing the toolkit, you should read the *Release Notes*, as they provide details on installation and software functionality.

> The driver and toolkit must be installed for CUDA to function. If you have not installed a stand-alone driver, install the driver from the NVIDIA CUDA Toolkit.

Graphical Installation

Install the CUDA Software by executing the CUDA installer and following the on-screen prompts.

Silent Installation

Alternatively, the installer can be executed in silent mode by executing the package with the **-s** flag. Additional flags can be passed which will install specific subpackages instead of all packages. Allowed subpackage names are: CUDAToolkit_6.5, CUDASamples_6.5, CUDAVisualStudioIntegration_6.5, and Display.Driver. For example, to install only the driver and the toolkit components:

```
<PackageName>.exe -s CUDAToolkit_6.5 Display.Driver
```

Subpackage Details

▶ Display Driver

Required to run CUDA applications.

▶ CUDA Toolkit

The CUDA Toolkit installation defaults to **C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v6.5**. 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 documentation, including:

- ▶ *CUDA C Programming Guide*
- ▶ *CUDA C Best Practices Guide*
- ▶ documentation for the CUDA libraries
- ▶ other CUDA Toolkit-related documentation

▶ CUDA Visual Studio Integration

The CUDA Visual Studio Integration registers the CUDA plugins with the supported and installed versions of Visual Studio on the system. If not on Windows XP, it will also install Nsight Visual Studio Edition. This integration allows for CUDA development within Visual Studio.

▶ CUDA Samples

The CUDA Samples contain source code for many example problems and templates with Microsoft Visual Studio 2010, 2012, and 2013 projects.

On Windows XP, the CUDA Samples installation defaults to `C:\Documents and Settings\All Users\Application Data\NVIDIA Corporation\CUDA Samples\v6.5`.

On Windows 7, Windows 8.1, Windows Server 2008 R2, and Windows Server 2012 R2, the CUDA Samples installation defaults to `C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5`.

> `C:\ProgramData\` is a hidden folder. It can be made visible within Windows Explorer by enabling it through the menu options within Windows Explorer (Tools | Options).

Extracting and Inspecting the Files Manually

Sometimes it may be desirable to extract or inspect the installable files directly, such as in enterprise deployment, or to browse the files before installation. The installation package can be extracted using a decompression tool which supports the LZMA compression method, such as 7-zip or WinZip.

Once extracted, the CUDA Toolkit files will be in the `CUDAToolkit` folder, and similarly for the CUDA Samples and CUDA Visual Studio Integration. Within each directory is a .dll and .nvi file that can be ignored as they are not part of the installable files.

> Accessing the files in this manner does not set up any environment settings, such as variables or Visual Studio integration. This is intended for enterprise-level deployment.

2.3.1. Uninstalling the CUDA Software

All subpackages can be uninstalled through the Windows Control Panel.

On Windows XP, use the Add or Remove Programs widget to select and remove individual subpackages.

On Windows 7, Windows 8.1, Windows Server 2008 R2, and Windows Server 2012 R2, use the Programs and Features widget to select and remove individual subpackages.

2.4. Use a Suitable Driver Model

On Windows 7 and later, the operating system provides two *driver models* under which the NVIDIA Driver may operate:

▶ The *WDDM* driver model is used for display devices.
▶ The *Tesla Compute Cluster (TCC)* mode of the NVIDIA Driver is available for non-display devices such as NVIDIA Tesla GPUs; it uses the Windows *WDM* driver model.

The TCC driver mode provides a number of advantages for CUDA applications on GPUs that support this mode. For example:

▶ TCC eliminates the timeouts that can occur when running under WDDM due to the Windows Timeout Detection and Recovery mechanism for display devices.
▶ TCC allows the use of CUDA with Windows Remote Desktop, which is not possible for WDDM devices.
▶ TCC allows the use of CUDA from within processes running as Windows services, which is not possible for WDDM devices.
▶ TCC reduces the latency of CUDA kernel launches.

TCC is enabled by default on most recent NVIDIA Tesla GPUs. To check which driver mode is in use and/or to switch driver modes, use the `nvidia-smi` tool that is included with the NVIDIA Driver installation (see `nvidia-smi -h` for details).

> Keep in mind that when TCC mode is enabled for a particular GPU, that GPU *cannot* be used as a display device.

> NVIDIA GeForce GPUs do not support TCC mode.

2.5. Verify the Installation

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

2.5.1. Running the Compiled 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

CUDA Samples include 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 **deviceQuery** program located here:

Windows XP:

```
C:\Documents and Settings\All Users\Application Data\NVIDIA Corporation\CUDA
 Samples\v6.5\bin\win32\Release
```

Windows 7, Windows 8.1, Windows Server 2008 R2, and Windows Server 2012 R2:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\bin\win32\Release
```

This assumes 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 deviceQuery CUDA Sample

The exact appearance and the output lines might be different on your system. The important outcomes are that a device was found, that the device(s) match what is installed in your system, and that the test passed.

If a CUDA-capable device and the CUDA Driver are installed but **deviceQuery** reports that no CUDA-capable devices are present, ensure the deivce and driver are properly installed.

Running the **bandwidthTest** program, located in the same directory as **deviceQuery** above, ensures that the system and the CUDA-capable device are able to communicate correctly. The output should resemble Figure 2.

Figure 2   Valid Results from bandwidthTest CUDA Sample

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-capable 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\CUDA Samples\v6.5\bin
\win32\Release
```

(or ...**\win64\Release** on 64-bit Windows)

▸   Windows 7, Windows 8.1, Windows Server 2008 R2, and Windows Server 2012 R2:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\bin\win32\Release
```

(or ...**\win64\Release** on 64-bit Windows)

Chapter 3.
COMPILING CUDA PROGRAMS

The project files in the CUDA Samples 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 2010, 2012, or 2013 (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

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5\<category>\<sample_name>
```

or the global solution files **Samples*.sln** located in

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v6.5
```

CUDA Samples are organized according to **<category>**. Each sample is organized into one of the following folders: (**0_Simple**, **1_Utilities**, **2_Graphics**, **3_Imaging**, **4_Finance**, **5_Simulations**, **6_Advanced**, **7_CUDALibraries**).

3.1. Compiling Sample Projects

The **bandwidthTest** project is a good sample project to build and run. It is located in the **NVIDIA Corporation\CUDA Samples\v6.5\1_Utilities\bandwidthTest** directory.

The output is placed in **CUDA Samples\v6.5\bin\win32\Release**. (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 2.

3.2. Sample Projects

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

A few of the example projects require some additional setup. The **simpleD3D9** example requires the system to have a Direct3D SDK (June 2010 or later) installed and the Visual

C++ directory paths (located in **Tools** > **Options...**) properly configured. Consult the Direct3D documentation for additional details.

These sample projects also make use of the `$CUDA_PATH` environment variable to locate where the CUDA Toolkit and the associated `.props` files are.

The environment variable is set automatically using the Build Customization `CUDA 6.5.props` file, and is installed automatically as part of the CUDA Toolkit installation process.

For Visual Studio 2010, the `CUDA 6.5.props` file is installed into:

```
C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V100\BuildCustomizations
```

For Visual Studio 2012, the `CUDA 6.5.props` file is installed into:

```
C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V110\BuildCustomizations
```

For Visual Studio 2013, the `CUDA 6.5.props` file is installed into:

```
C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V120\BuildCustomizations
```

You can reference this `CUDA 6.5.props` file when building your own CUDA applications.

3.3. Build Customizations for New Projects

When creating a new CUDA application, the Visual Studio project file must be configured to include CUDA build customizations. To accomplish this, click File-> New | Project... NVIDIA-> CUDA->, then select a template for your CUDA Toolkit version. For example, selecting the "CUDA 6.5 Runtime" template will configure your project for use with the CUDA 6.5 Toolkit. The new project is technically a C++ project (.vcxproj) that is preconfigured to use NVIDIA's Build Customizations. All standard capabilities of Visual Studio C++ projects will be available.

To specify a custom CUDA Toolkit location, under **CUDA C/C++**, select **Common**, and set the **CUDA Toolkit Custom Dir** field as desired. Note that the selected toolkit must match the version of the Build Customizations.

3.4. Build Customizations for Existing Projects

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

1. Open the Visual Studio 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 `$(CUDA_PATH)` . Note that the `$(CUDA_PATH)` environment variable is set by the installer.

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.

If you use the **$(CUDA_PATH)** environment variable to target a version of the CUDA Toolkit for building, and you perform an installation or uninstallation of any version of the CUDA Toolkit, you should validate that the **$(CUDA_PATH)** environment variable points to the correct installation directory of the CUDA Toolkit for your purposes. You can access the value of the **$(CUDA_PATH)** environment variable via the following steps:

1. Open a run window from the Start Menu
2. Run:
   ```
   control sysdm.cpl
   ```
3. Select the "Advanced" tab at the top of the window
4. Click "Environment Variables" at the bottom of the window

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 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/"
```

Chapter 4.
ADDITIONAL CONSIDERATIONS

Now that you have CUDA-capable hardware and the NVIDIA CUDA Toolkit 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.

A number of helpful development tools are included in the CUDA Toolkit or are available for download from the NVIDIA Developer Zone to assist you as you develop your CUDA programs, such as NVIDIA® Nsight™ Visual Studio Edition, NVIDIA Visual Profiler, and cuda-memcheck.

For technical support on programming questions, consult and participate in the developer forums at http://developer.nvidia.com/cuda/.