NVIDIA CUDA C GETTING STARTED GUIDE FOR MAC OS X

DU-05348-001_v04 | March 2011

Installation and Verification on Mac OS X

DOCUMENT CHANGE HISTORY

DU-05348-001_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	December 22, 2010	VS	Fixed typo on page 7, section entitled “Running the Binaries” Corrected: "DYLD_LIBRARY_PATH"
04	March 3, 2011	CW	Updated for CUDA Toolkit 4.0

TABLE OF CONTENTS

LIST OF FIGURES

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 and C for CUDA, 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
- Mac OS X v. 10.5.6 or later (10.6.3 or later for 64-bit CUDA applications)
- The gcc compiler and toolchain installed using Xcode
- CUDA software (available at no cost from http://www.nvidia.com/cuda)

ABOUT THIS DOCUMENT

This document is intended for readers familiar with the Mac OS X environment and the compilation of C programs from the command line. 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 Mac OS X consists of three simple steps:

- Verify the system has a CUDA-enabled GPU, a supported version of Mac OS X, and that gcc has been installed via Xcode.
- Download the CUDA driver and software.
- Install the CUDA driver and 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, and 500 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 Mac OS X system uses, under the **Apple** menu select **About This Mac**, click the **More Info ...** button, and then select **Graphics/Displays** under the **Hardware** list.

VERIFY THE CORRECT VERSION OF MAC OS X

The CUDA Development Tools require an Intel-based Mac running Mac OS X v. 10.5.6 or later. For 64-bit CUDA applications, Mac OS X v. 10.6.3 or later is required. To check which version you have, go to the Apple menu on the desktop and select **About This Mac**. You should see a dialog box similar to Figure 1.

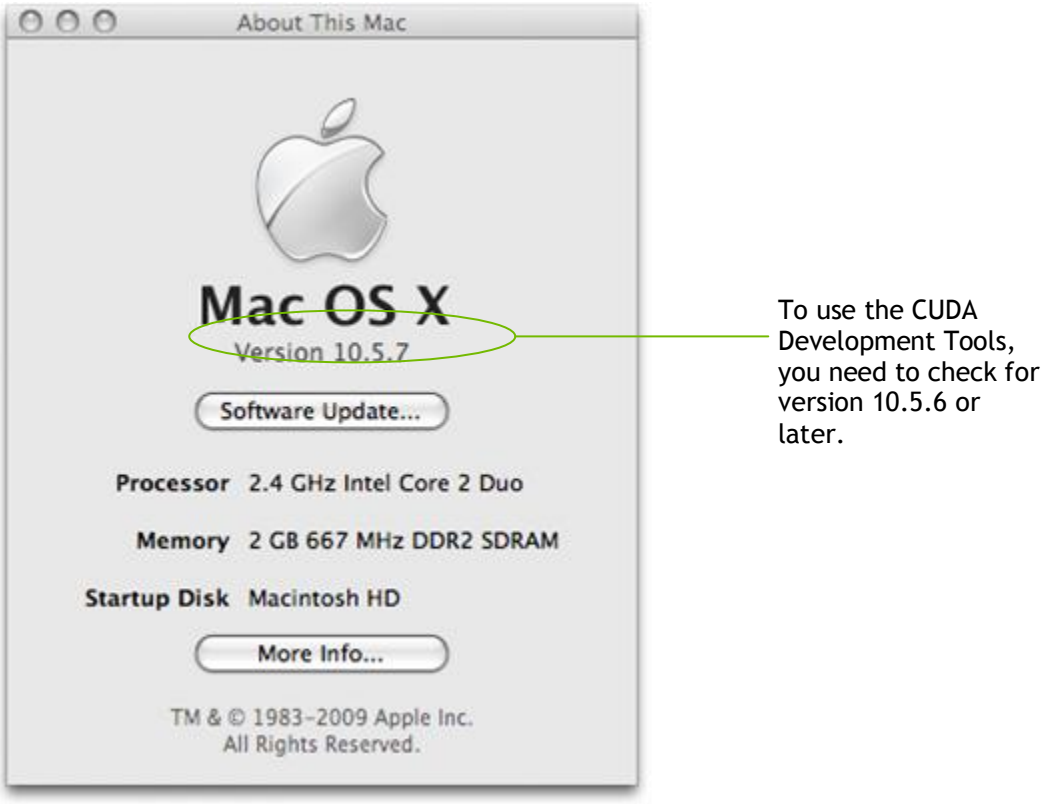

Figure 1. About This Mac Dialog Box

Verify That gcc Is Installed

The **gcc** compiler and toolchain are installed using the installation of Xcode. The Xcode development environment is found on the **Xcode Developer Tools DVD** that ships with new Mac systems and with Leopard, if you buy the operating-system upgrade. When installing Xcode, the package that contains **gcc** and the necessary tools is called *Developer Tools Essentials*. You can verify that **gcc** is installed entering the command **`/usr/bin/gcc --help`** from a Terminal window.

DOWNLOAD THE CUDA SOFTWARE

Once you have verified that you have a supported NVIDIA processor and a supported version the MAX OS, you need to download the CUDA software.

The CUDA software is available at no cost from the main CUDA download site at http://www.nvidia.com/object/cuda_get.html .

Download the following packages for the latest version of the Development Tools:

- CUDA Driver
- CUDA Toolkit
- GPU Computing SDK

The CUDA Toolkit contains the tools needed to compile and build a CUDA application in conjunction with the **nvcc** compilation driver. It includes tools, libraries, header files, and other resources.

The GPU Computing SDK includes sample projects that provide source code and other resources for constructing CUDA programs.

INSTALL THE CUDA DRIVER AND SOFTWARE

Use the following procedure to successfully install the CUDA driver and software. For information not listed here, see the documentation under **`/usr/local/cuda/doc`** in the download location.

Before installing the CUDA software packages, you should read the **Release Notes** bundled with each, as those notes provide important details on installation and software functionality.

Then, follow these few steps for a successful installation.

1. **Uninstall any previous versions of the CUDA Toolkit and GPU Computing SDK.** Do this by deleting the files from **`/usr/local/cuda`** and from **`/Developer/GPU Computing`**, the default installation locations. (Note that older versions of the SDK installed into **`/Developer/CUDA`** by default rather than **`/Developer/GPU Computing`**.) Adjust accordingly if you placed the files in non-default directories. (If you wish to keep the files so you can compile for different versions of CUDA software, then rename the existing directories before installing the new version and modify your **`Makefile`** accordingly.)

2. **Install the CUDA Driver.**
   Install the CUDA driver package by executing the installer and following the on-screen prompts. This will install **/Library/Frameworks/CUDA.framework** and the UNIX-compatibility stub **/usr/local/cuda/lib/libcuda.dylib** that refers to it.
3. **Install the CUDA Toolkit.**
   Install the CUDA Toolkit by executing the Toolkit installer package and following the on-screen prompts. The CUDA Toolkit supplements the CUDA Driver with compilers and additional libraries and header files that are installed into **/usr/local/cuda** by default.
4. **Define the environment variables.**
   - The **PATH** variable needs to include **/usr/local/cuda/bin**.
   - **DYLD_LIBRARY_PATH** needs to contain **/usr/local/cuda/lib**.

   The typical way to place these values in your environment is with the following commands:

   ```
   export PATH=/usr/local/cuda/bin:$PATH
   export DYLD_LIBRARY_PATH=/usr/local/cuda/lib:$DYLD_LIBRARY_PATH
   ```

   To make these settings permanent, place them in **~/.bash_profile**.
5. **Install the CUDA SDK.**
   The installation process places the files in **/Developer/GPU Computing**.

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.

Verify the Driver Installation

If the CUDA Driver is installed correctly, the CUDA kernel extension (`/System/Library/Extensions/CUDA.kext`) should be loaded automatically at boot time. To verify that it is loaded, use the command **kextstat | grep -i cuda** .

Compiling the Examples

The version of the CUDA Toolkit can be checked by running **nvcc -V** in a terminal window. The **nvcc** command runs the compiler driver that compiles CUDA programs. It calls the **gcc** compiler for C code and the NVIDIA PTX compiler for the CUDA code.

NVIDIA includes sample programs in source form in the GPU Computing SDK. You should compile them all by changing to **/Developer/GPU Computing/C** and type **make**. The resulting binaries will be installed under the home directory in **/Developer/GPU Computing/C/bin/darwin/release**.

Running the Binaries

The sample projects use libraries pointed to by **DYLD_LIBRARY_PATH**, as described earlier, so make sure it points to the right directory.

The executables need to find **libcutil.a** in **/Developer/GPU Computing/C/lib** and the corresponding include file **in /Developer/GPU Computing/C/common/inc**. They also need to access graphics libraries in **/Developer/GPU Computing/C/common/lib/darwin**. If you have installed the CUDA software as explained in this document, these locations are defaults and the programs should run without difficulty.

Once these required files are in place, go to **/Developer/GPU Computing/C/bin/darwin/release** and run **deviceQuery**. If CUDA is installed and configured correctly, the output for **deviceQuery** should look similar to Figure 2.

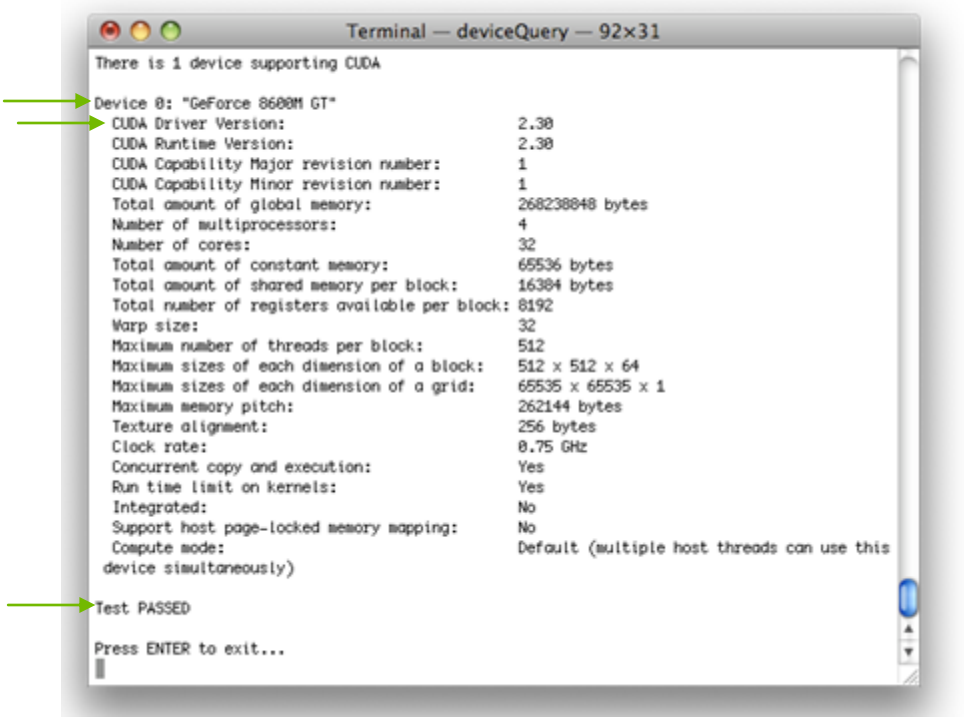

Figure 2. Valid Results from Sample CUDA deviceQuery Program

Note that the parameters for your CUDA device will vary. The key lines are the first and second ones that confirm a device was found and what model it is. Also, the next-to-last line, as indicated, should show that the test passed.

Running the **`bandwidthTest`** program ensures that the system and the CUDA device are able to communicate correctly. Figure 3 shows a sample of the output.

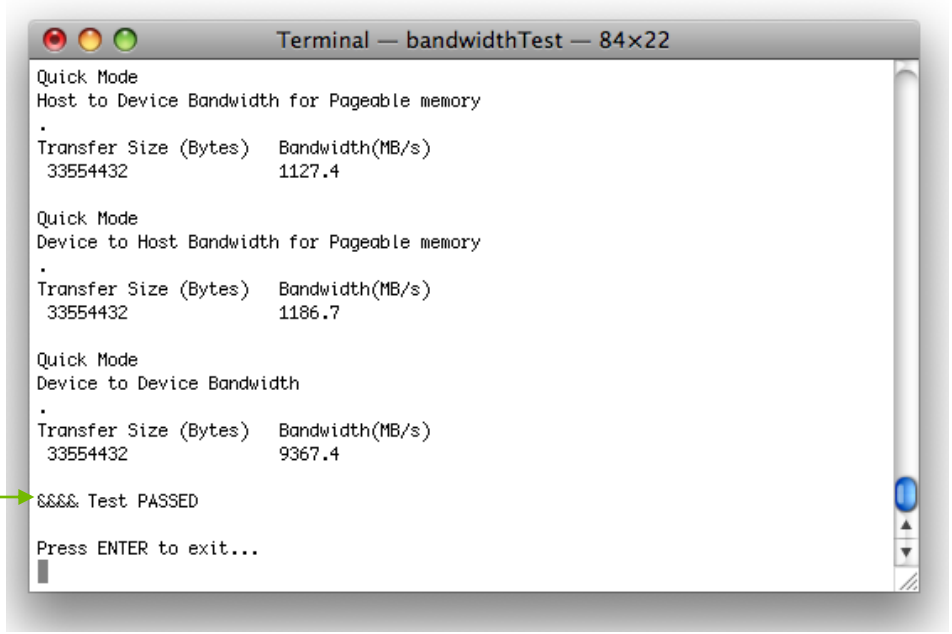

Figure 3. Valid Results from Sample CUDA bandwidthTest Program

Note that the measurements for your CUDA-enabled device description will vary from system to system. The important point is that you obtain measurements, and that the second-to-last line (in Figure 3) confirms that all necessary tests passed.

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

If you run into difficulties with the link step (such as libraries not being found), consult the Release Notes found in the doc folder in the SDK directory.

To see a graphical representation of what CUDA can do, run the sample particles executable.

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 **`/usr/local/cuda/doc`**.

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