



CUDA Installation



Jen-Hao Chen

March 28, 2016

CUDA Installation

Goal :

▶ **Win 10**

+ **Visual Studio community 2013**

+ **CUDA 7.5**

▶ **OR : Win 7 + VS 2010 + CUDA 6.5**



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



1. System Requirements

Table 1. Windows Operating System Support in CUDA 7.5

Operating System	Native x86_64	Cross (x86_32 on x86_64)
Windows 10	YES	YES
Windows 8.1	YES	YES
Windows 7	YES	YES
Windows Server 2012 R2	YES	YES
Windows Server 2008 R2	YES	YES

Table 2. Windows Compiler Support in CUDA 7.5

Compiler	IDE	Native x86_64	Cross (x86_32 on x86_64)
Visual C++ 12.0	Visual Studio 2013	YES	YES
	Visual Studio Community 2013	YES	NO
Visual C++ 11.0	Visual Studio 2012	YES	YES
Visual C++ 10.0 DEPRECATED	Visual Studio 2010	YES	YES

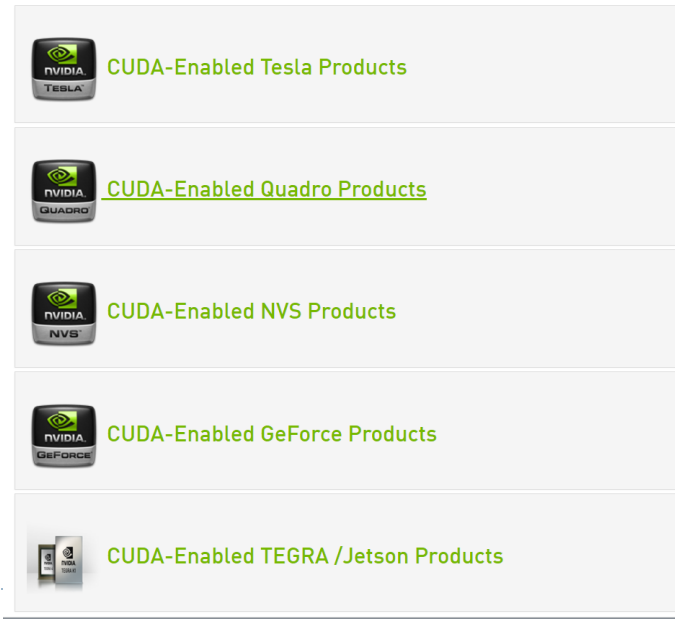


2. Installing CUDA Development Tools






Verify the system has a CUDA-capable GPU

- ▶ If you have an NVIDIA card that is listed in <http://developer.nvidia.com/cuda-gpus>, that GPU is CUDA-capable.

If you have an older NVIDIA GPU you may find it listed on our [legacy CUDA GPUs page](#)
Click the sections below to expand



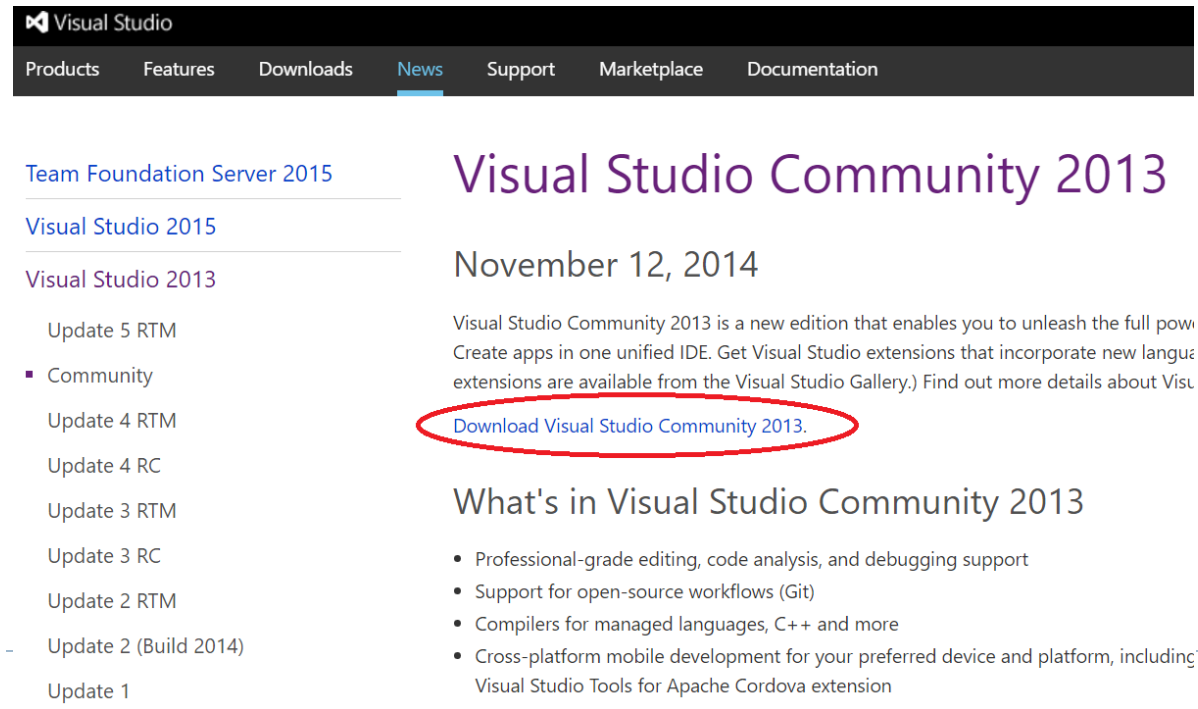
The screenshot displays a vertical list of five expandable sections, each with an NVIDIA logo icon and a title. The sections are:

-  [CUDA-Enabled Tesla Products](#)
-  [CUDA-Enabled Quadro Products](#)
-  [CUDA-Enabled NVS Products](#)
-  [CUDA-Enabled GeForce Products](#)
-  [CUDA-Enabled TEGRA /Jetson Products](#)

2. Installing CUDA Development Tools

Step.1 Install Visual Studio community 2013 (free)

- ▶ Download from <https://www.visualstudio.com/en-us/news/vs2013-community-vs.aspx>



The screenshot shows the Visual Studio website's news section. The navigation bar includes links for Products, Features, Downloads, News (highlighted), Support, Marketplace, and Documentation. On the left, a sidebar lists various Visual Studio versions and updates, with 'Visual Studio 2013' selected. The main content area features the title 'Visual Studio Community 2013' and the date 'November 12, 2014'. A paragraph of text describes the new edition, and a link 'Download Visual Studio Community 2013.' is circled in red. Below this, a section titled 'What's in Visual Studio Community 2013' lists several features.

Visual Studio

Products Features Downloads **News** Support Marketplace Documentation

Team Foundation Server 2015

Visual Studio 2015

Visual Studio 2013

- Update 5 RTM
- Community
- Update 4 RTM
- Update 4 RC
- Update 3 RTM
- Update 3 RC
- Update 2 RTM
- Update 2 (Build 2014)
- Update 1

Visual Studio Community 2013

November 12, 2014

Visual Studio Community 2013 is a new edition that enables you to unleash the full power of the Visual Studio IDE. (Create apps in one unified IDE. Get Visual Studio extensions that incorporate new languages and features that are available from the Visual Studio Gallery.) Find out more details about Visual Studio Community 2013.

[Download Visual Studio Community 2013.](#)

What's in Visual Studio Community 2013

- Professional-grade editing, code analysis, and debugging support
- Support for open-source workflows (Git)
- Compilers for managed languages, C++ and more
- Cross-platform mobile development for your preferred device and platform, including Visual Studio Tools for Apache Cordova extension

2. Installing CUDA Development Tools

Step.2 Download driver that supports CUDA 7.5

- ▶ Search keywords <GPU name, driver and CUDA 7.5>

or go to

NVIDIA > 驅動程式下載

NVIDIA 驅動程式下載

選項 1: 手動尋找合適的驅動程式。

產品類型: GeForce

產品系列: GeForce 900 Series

產品家族: GeForce GTX TITAN X

作業系統: Windows 10 32-bit

語言: Chinese (Traditional)

搜尋

選項 2: 自動尋找合適的驅動程式。

瞭解更多資訊

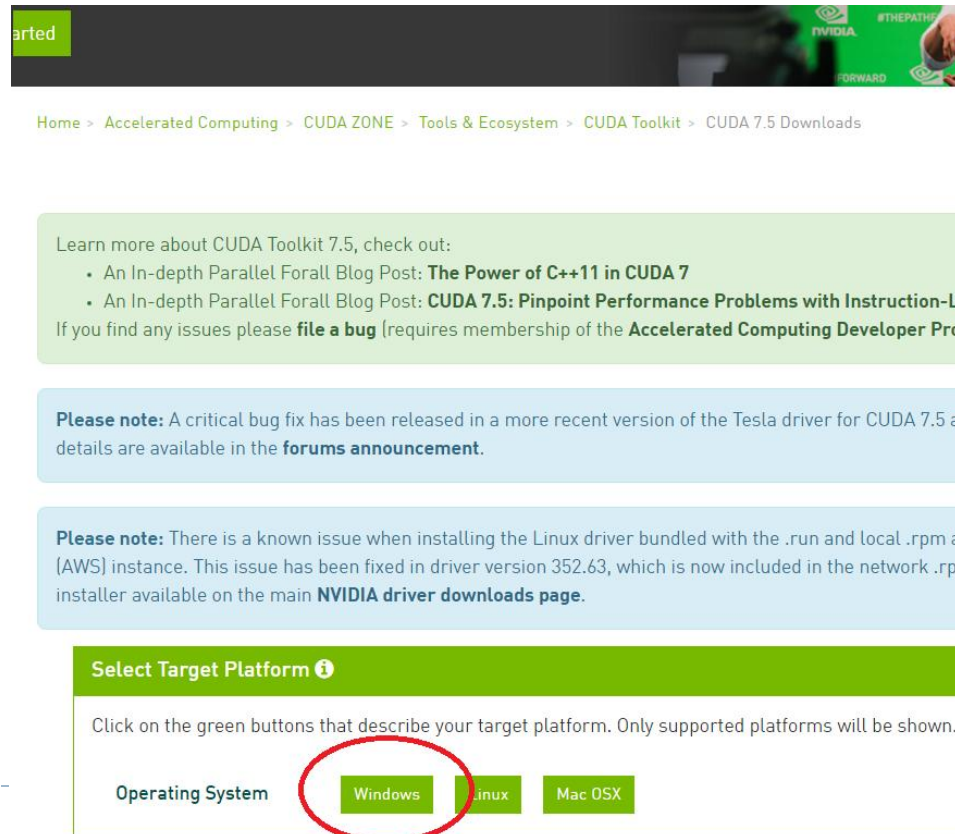
繪圖卡驅動程式 主機板驅動程式

更多的軟體和驅動程式

2. Installing CUDA Development Tools

Step.3 Download and install CUDA 7.5

► <https://developer.nvidia.com/cuda-downloads>



Started

Home > Accelerated Computing > CUDA ZONE > Tools & Ecosystem > CUDA Toolkit > CUDA 7.5 Downloads

Learn more about CUDA Toolkit 7.5, check out:

- An In-depth Parallel Forall Blog Post: **The Power of C++11 in CUDA 7**
- An In-depth Parallel Forall Blog Post: **CUDA 7.5: Pinpoint Performance Problems with Instruction-L**

If you find any issues please **file a bug** (requires membership of the **Accelerated Computing Developer Pro**

Please note: A critical bug fix has been released in a more recent version of the Tesla driver for CUDA 7.5 and details are available in the **forums announcement**.

Please note: There is a known issue when installing the Linux driver bundled with the .run and local .rpm on an AWS (AWS) instance. This issue has been fixed in driver version 352.63, which is now included in the network .rpm installer available on the main **NVIDIA driver downloads page**.

Select Target Platform ⓘ

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows Linux Mac OSX

3. Verify the Installation

Running the Compiled Examples

- ▶ To verify a correct configuration of the hardware and software, it is highly recommended that you run the **deviceQuery** program located at

C:\ProgramData\NVIDIA Corporation\CUDA

Samples\v7.5\1_Uutilities\deviceQuery

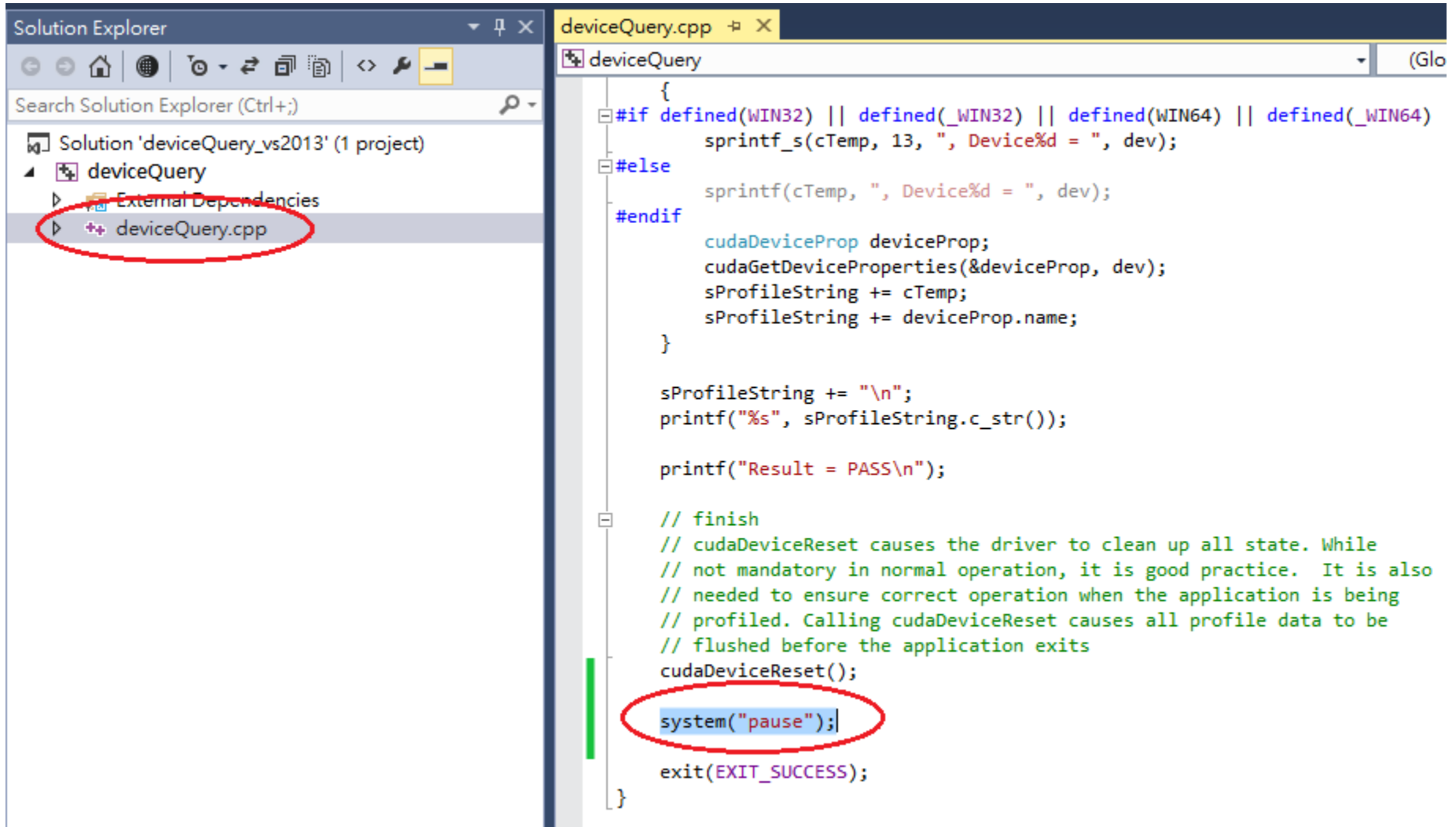


本機 > Win7 (C:) > ProgramData > NVIDIA Corporation > CUDA Samples > v7.5 > 1_Uutilities > deviceQuery

Print 列印

名稱	修改日期	類型	大小
deviceQuery.cpp	2015/5/27 下午 1...	C++ Source File	13 KB
deviceQuery_vs2010.sln	2015/8/16 下午 0...	Microsoft Visual ...	1 KB
deviceQuery_vs2010.vcxproj	2015/8/16 下午 0...	VC++ Project	5 KB
deviceQuery_vs2012.sln	2015/8/16 下午 0...	Microsoft Visual ...	1 KB
deviceQuery_vs2012.vcxproj	2015/8/16 下午 0...	VC++ Project	5 KB
deviceQuery_vs2013.sln	2015/8/16 下午 0...	Microsoft Visual ...	1 KB
deviceQuery_vs2013.vcxproj	2015/8/16 下午 0...	VC++ Project	5 KB
readme.txt	2015/8/16 下午 0...	文字文件	1 KB

3. Verify the Installation



The image shows a screenshot of Visual Studio with the Solution Explorer on the left and the code editor on the right. The Solution Explorer shows the project 'deviceQuery' with the file 'deviceQuery.cpp' selected. The code editor shows the implementation of the deviceQuery function, which prints the device name and profile string, and includes a system('pause') call to verify the installation.

```
deviceQuery.cpp  
deviceQuery  
{  
    #if defined(WIN32) || defined(_WIN32) || defined(WIN64) || defined(_WIN64)  
        sprintf_s(cTemp, 13, " Device%d = ", dev);  
    #else  
        sprintf(cTemp, " Device%d = ", dev);  
    #endif  
    cudaDeviceProp deviceProp;  
    cudaGetDeviceProperties(&deviceProp, dev);  
    sProfileString += cTemp;  
    sProfileString += deviceProp.name;  
}  
sProfileString += "\n";  
printf("%s", sProfileString.c_str());  
  
printf("Result = PASS\n");  
  
// finish  
// cudaDeviceReset causes the driver to clean up all state. While  
// not mandatory in normal operation, it is good practice. It is also  
// needed to ensure correct operation when the application is being  
// profiled. Calling cudaDeviceReset causes all profile data to be  
// flushed before the application exits  
cudaDeviceReset();  
system("pause");  
exit(EXIT_SUCCESS);  
}
```

Valid Results from **deviceQuery** CUDA Sample (**Quadro K600**)

```

CUDA Device Query (Runtime API) version (CUDART static linking)

Detected 1 CUDA Capable device(s)

Device 0: "Quadro 600"
  CUDA Driver Version / Runtime Version      7.5 / 7.5
  CUDA Capability Major/Minor version number: 2.1
  Total amount of global memory:            1024 MBytes (1073741824 bytes)
  ( 2) Multiprocessors, ( 48) CUDA Cores/MP: 96 CUDA Cores
  GPU Max Clock rate:                       1280 MHz (1.28 GHz)
  Memory Clock rate:                        800 Mhz
  Memory Bus Width:                         128-bit
  L2 Cache Size:                            131072 bytes
  Maximum Texture Dimension Size (x,y,z)    1D=(65536), 2D=(65536, 65535), 3D=(2048, 2048, 2048)
  Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers
  Total amount of constant memory:          65536 bytes
  Total amount of shared memory per block:   49152 bytes
  Total number of registers available per block: 32768
  Warp size:                                32
  Maximum number of threads per multiprocessor: 1536
  Maximum number of threads per block:      1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size (x,y,z): (65535, 65535, 65535)
  Maximum memory pitch:                    2147483647 bytes
  Texture alignment:                       512 bytes
  Concurrent copy and kernel execution:     Yes with 1 copy engine(s)
  Run time limit on kernels:                Yes
  Integrated GPU sharing Host Memory:       No
  Support host page-locked memory mapping:  Yes
  Alignment requirement for Surfaces:       Yes
  Device has ECC support:                   Disabled
  CUDA Device Driver Mode (TCC or WDDM):    WDDM (Windows Display Driver Model)
  Device supports Unified Addressing (UVA): Yes
  Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
  Compute Mode:
    < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 7.5, CUDA Runtime Version = 7.5, NumDevs = 1, Device0 = Quadro 600
Result = PASS
```

Valid Results from **deviceQuery** CUDA Sample (**Quadro M4000**)

```
CUDA Device Query (Runtime API) version (CUDA static linking)
```

```
Detected 1 CUDA Capable device(s)
```


```
Device 0: "Quadro M4000"
```

```
CUDA Driver Version / Runtime Version      7.5 / 7.5
CUDA Capability Major/Minor version number: 5.2
Total amount of global memory:             8192 MBytes (8589934592 bytes)
(13) Multiprocessors, (128) CUDA Cores/MP: 1664 CUDA Cores
GPU Max Clock rate:                        773 MHz (0.77 GHz)
Memory Clock rate:                         3005 Mhz
Memory Bus Width:                          256-bit
L2 Cache Size:                             2097152 bytes
Maximum Texture Dimension Size (x,y,z)     1D=(65536), 2D=(65536, 65536), 3D=(4096, 4096, 4096)
Maximum Layered 1D Texture Size, (num) layers 1D=(16384), 2048 layers
Maximum Layered 2D Texture Size, (num) layers 2D=(16384, 16384), 2048 layers
Total amount of constant memory:           65536 bytes
Total amount of shared memory per block:    49152 bytes
Total number of registers available per block: 65536
Warp size:                                  32
Maximum number of threads per multiprocessor: 2048
Maximum number of threads per block:       1024
Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
Max dimension size of a grid size (x,y,z): (2147483647, 65535, 65535)
Maximum memory pitch:                      2147483647 bytes
Texture alignment:                         512 bytes
Concurrent copy and kernel execution:       Yes with 2 copy engine(s)
Run time limit on kernels:                 Yes
Integrated GPU sharing Host Memory:         No
Support host page-locked memory mapping:    Yes
Alignment requirement for Surfaces:         Yes
Device has ECC support:                    Disabled
CUDA Device Driver Mode (TCC or WDDM):      WDDM (Windows Display Driver Model)
Device supports Unified Addressing (UVA):   Yes
Device PCI Domain ID / Bus ID / location ID: 0 / 1 / 0
Compute Mode:
  < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >
```

```
deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 7.5, CUDA Runtime Version = 7.5, NumDevs = 1, Device0 = Quadro M4000
Result = PASS
```

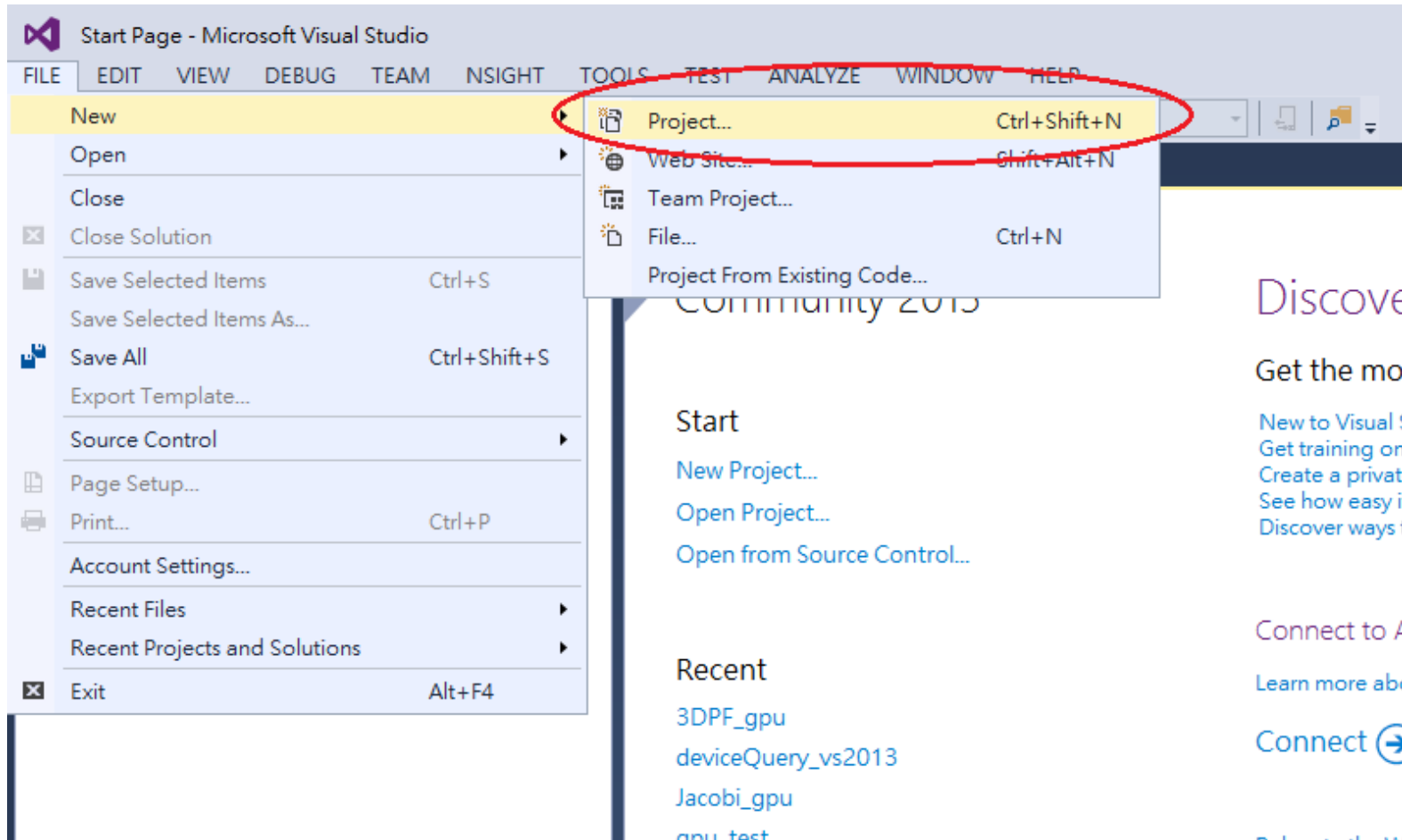
Compiling CUDA Programs

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 7.5 Runtime” template** will configure your project for use with the CUDA 7.5 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.**
-
- 

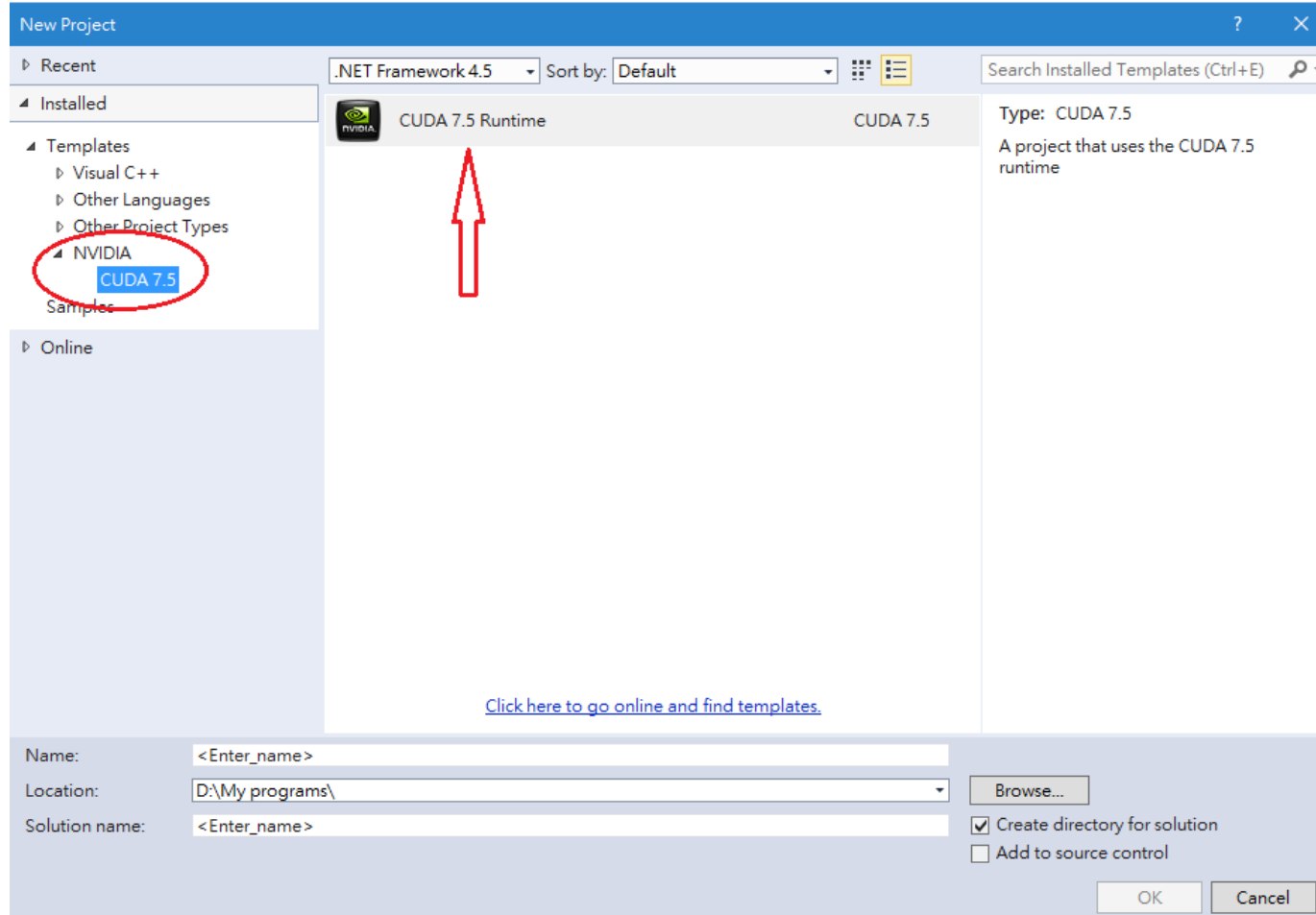
Compiling CUDA Programs

Build Customizations for New Projects



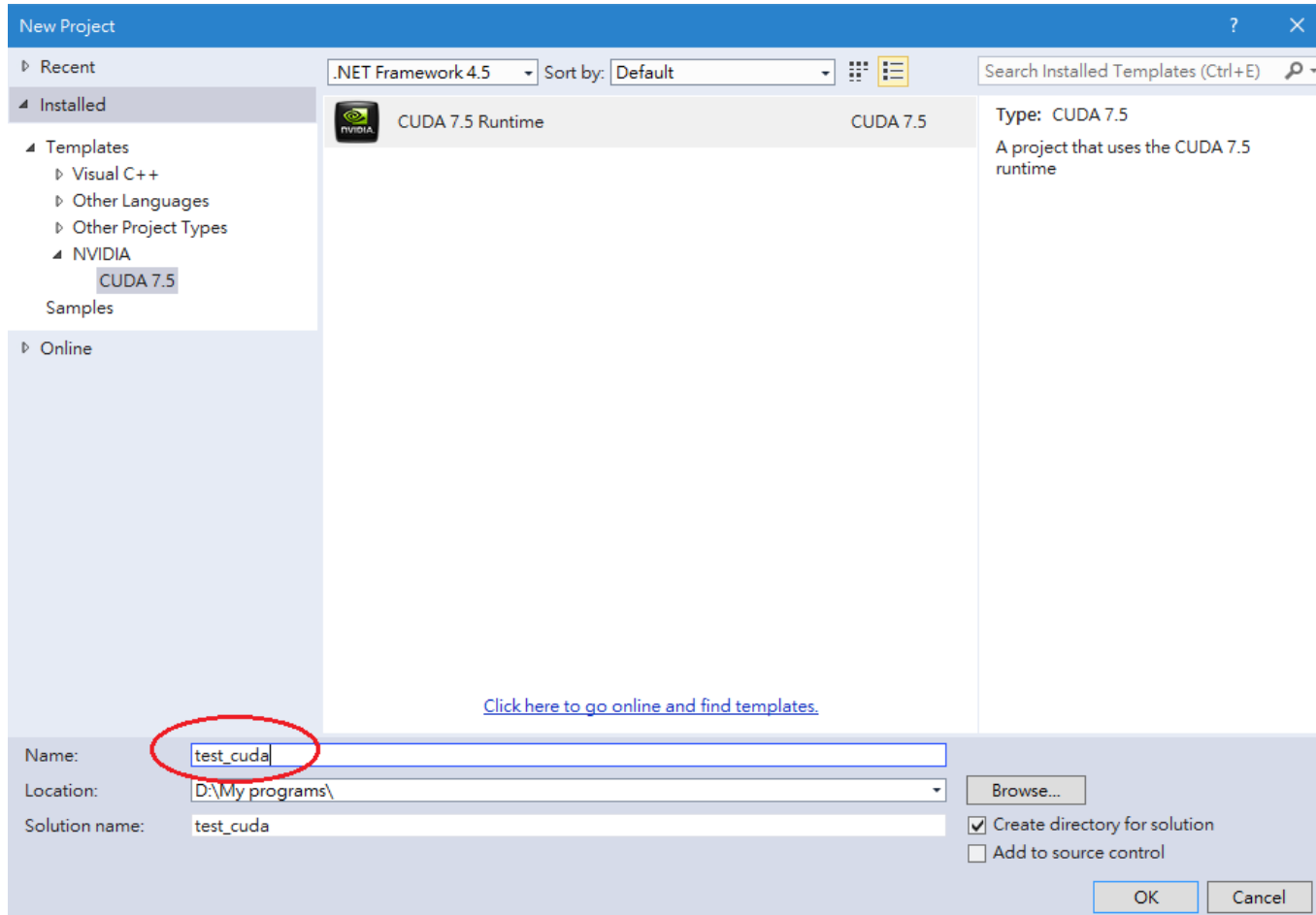
Compiling CUDA Programs

Build Customizations for New Projects



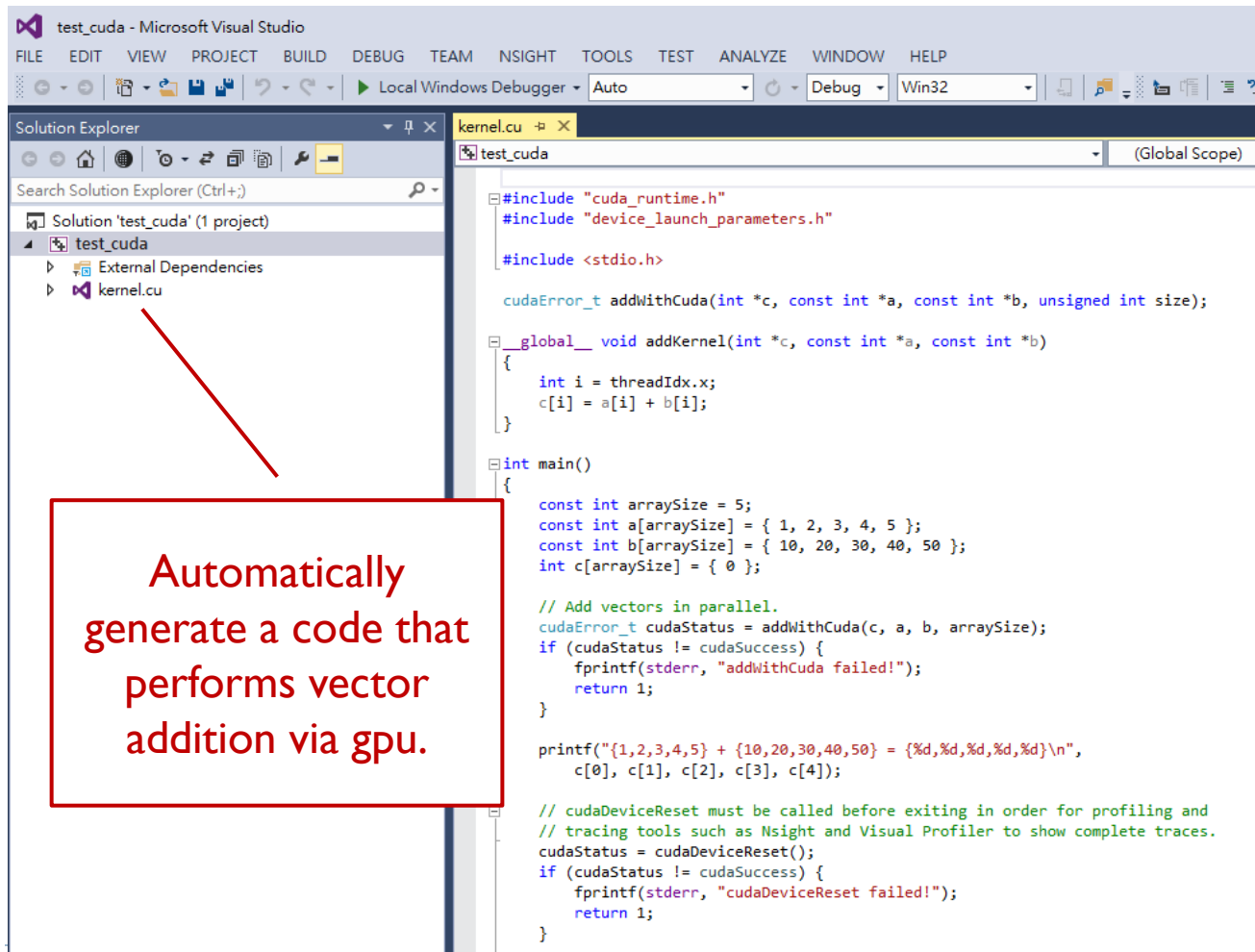
Compiling CUDA Programs

Build Customizations for New Projects



Compiling CUDA Programs

Build Customizations for New Projects



test_cuda - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM NSIGHT TOOLS TEST ANALYZE WINDOW HELP

Local Windows Debugger Auto Debug Win32

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'test_cuda' (1 project)

- test_cuda
 - External Dependencies
 - kernel.cu

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

cudaError_t addWithCuda(int *c, const int *a, const int *b, unsigned int size);

__global__ void addKernel(int *c, const int *a, const int *b)
{
    int i = threadIdx.x;
    c[i] = a[i] + b[i];
}

int main()
{
    const int arraySize = 5;
    const int a[arraySize] = { 1, 2, 3, 4, 5 };
    const int b[arraySize] = { 10, 20, 30, 40, 50 };
    int c[arraySize] = { 0 };

    // Add vectors in parallel.
    cudaError_t cudaStatus = addWithCuda(c, a, b, arraySize);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "addWithCuda failed!");
        return 1;
    }

    printf("{1,2,3,4,5} + {10,20,30,40,50} = {%d,%d,%d,%d,%d}\n",
        c[0], c[1], c[2], c[3], c[4]);

    // cudaDeviceReset must be called before exiting in order for profiling and
    // tracing tools such as Nsight and Visual Profiler to show complete traces.
    cudaStatus = cudaDeviceReset();
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "cudaDeviceReset failed!");
        return 1;
    }
}
```

Automatically generate a code that performs vector addition via gpu.

Compiling CUDA Programs

Build Customizations for New Projects

test_cuda - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM NSIGHT TOOLS TEST ANALYZE WINDOW

Local Windows Debugger Auto Debug

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution 'test_cuda' (1 project)

- test_cuda
 - External Dependencies
 - kernel.cu

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

int main()
{
    return 0;
}
```

Delete all and keep these two header files.

Remember that these two header files should be put somewhere that gpu codes can see.

An example: 3DPF_gpu

The screenshot displays the Microsoft Visual Studio interface for the 3DPF_gpu project. The Solution Explorer on the left shows the project structure, with the '3DPF_gpu' folder expanded. A red circle highlights the 'External Dependencies' folder, which contains the following files:

- C3DPNP.cxx
- C3DPNP.hpp
- CGlobal.cxx
- CGlobal.hpp
- CSolver.cu
- CSolver.hpp
- GJacobi.cu
- GJacobi.hpp
- kernel.cu

The main editor window shows the source code for '3DPF_gpu'. The code includes a list of protein structures and their corresponding table IDs and dates:

```
13. Ca3: 3DPF-Table3.39, 2014.7.4.  
14. Ca3: 3DPF-Table3.48, 2014.11.12.  
15. GA2: 3DPF-Table3.52, 2015.3.9.  
16. TRPV: 3DPF-Table3.54, 2015.5.6.  
*/
```

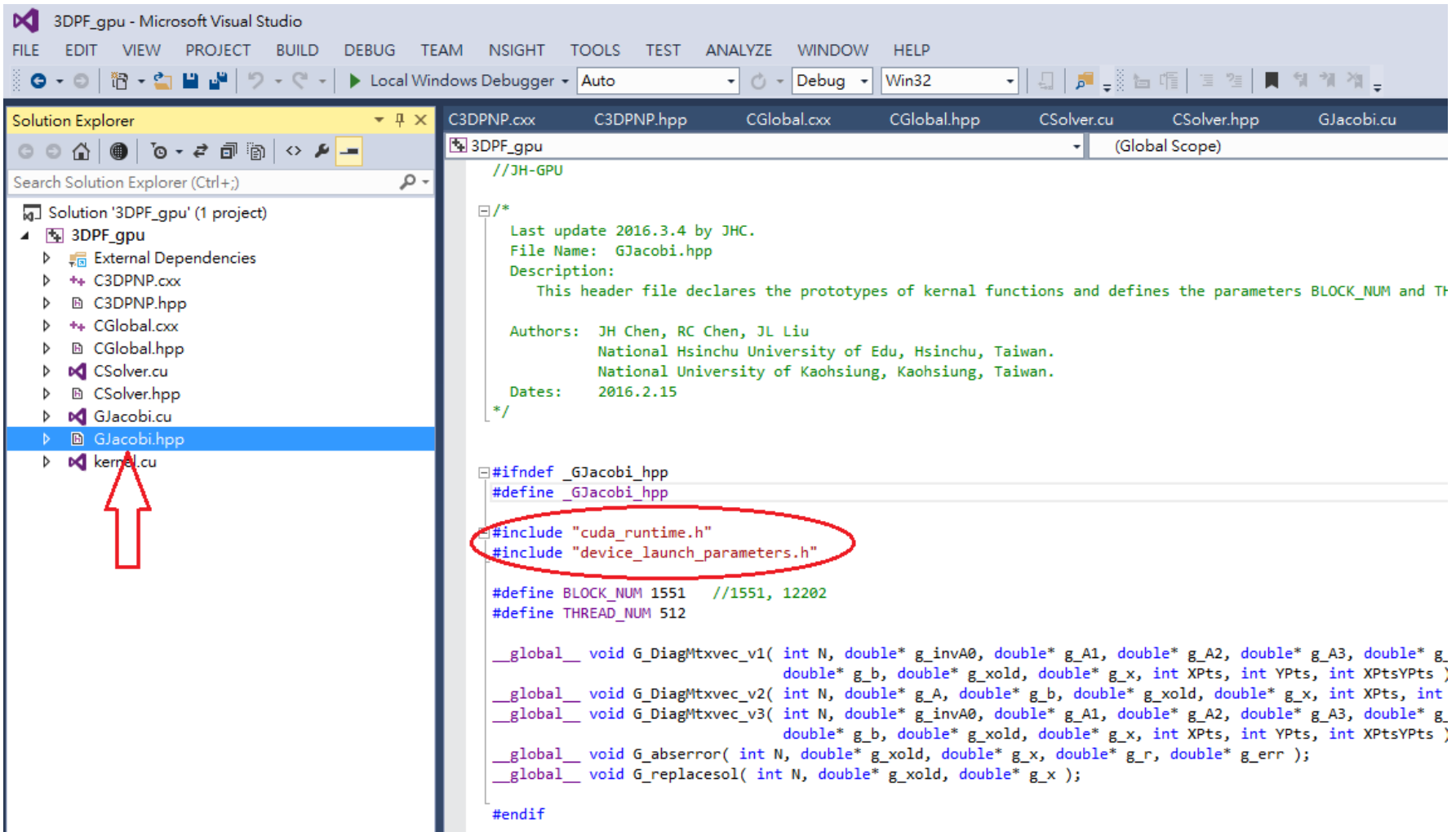
The code also includes the following header and main function:

```
#include "C3DPNP.hpp"  
  
int main(void) //JH-GPU  
{  
    C3DPNP wk;  
  
    // NCX: Ex3.56, Mol, 2015.5.20.  
    wk.Startup_NCX(); // 2015.5.20; Set Global Values Physical and Geometrical Parameter  
    wk.CreatePtr(); // Create Pointer (Dynamic) Data Members, 2011.7.5.  
    wk.ReadAtomPQRData_NCX(); // 2015.5.20; Read AtomPQR_TRPV.txt and set atom data (SN,  
    wk.ReadNaCaChannel_NCX(); // 2015.5.21, Read XYZR in ChannelNa.txt and ChannelCa.txt  
    wk.BindPhiStrc_NCX(); // 2015.5.21, Electric and Steric Potentials in NCX Binding Si  
    wk.CutProtein_NCX(); // 5/23; 2015.4.23, Cut to a Smaller Protein for Quick Solution  
    wk.MeshGen_NCX(); // 2015.5.21.  
    wk.ChanArea_NCX(); // 5/23; 2015.4.17; 2013.8.7; Calculate channel area at any Z cro  
    //wk.GummellLoop_Mol_NCX(); // 2015.5.23; 10/12; R2.75, 2014.3.1, Molecular-Continuum  
    wk.GummellLoop_SMIB_NCX(); // 2015.5.27; 2015.1.25; T4.5, 2012.1.16; Nonlinear PNP, 2  
  
    wk.DeletePtr(); // Delete Pointer Data Members, 2011.7.5.  
    system("PAUSE");  
    return 0;  
}
```

The code also includes a comment block at the bottom:

```
/*  
NCX: NCX Channel (real protein structure), 2015.5.17.  
CaVab: CaVab Channel (real protein structure), 2015.4.22.  
TRPV: TRPV1 Channel (real protein structure). 2015.4.17.
```

An example: 3DPF_gpu



The image shows a screenshot of Microsoft Visual Studio with the 3DPF_gpu project open. The Solution Explorer on the left shows the project structure, with GJacobi.hpp selected. A red arrow points to GJacobi.hpp in the Solution Explorer. The main editor window displays the contents of GJacobi.hpp, which includes CUDA runtime and device launch parameters headers. A red circle highlights the #include statements for 'cuda_runtime.h' and 'device_launch_parameters.h'.

```
//JH-GPU
/*
Last update 2016.3.4 by JHC.
File Name: GJacobi.hpp
Description:
This header file declares the prototypes of kernal functions and defines the parameters BLOCK_NUM and TH

Authors:  JH Chen, RC Chen, JL Liu
National Hsinchu University of Edu, Hsinchu, Taiwan.
National University of Kaohsiung, Kaohsiung, Taiwan.

Dates:   2016.2.15
*/

#ifndef _GJacobi_hpp
#define _GJacobi_hpp

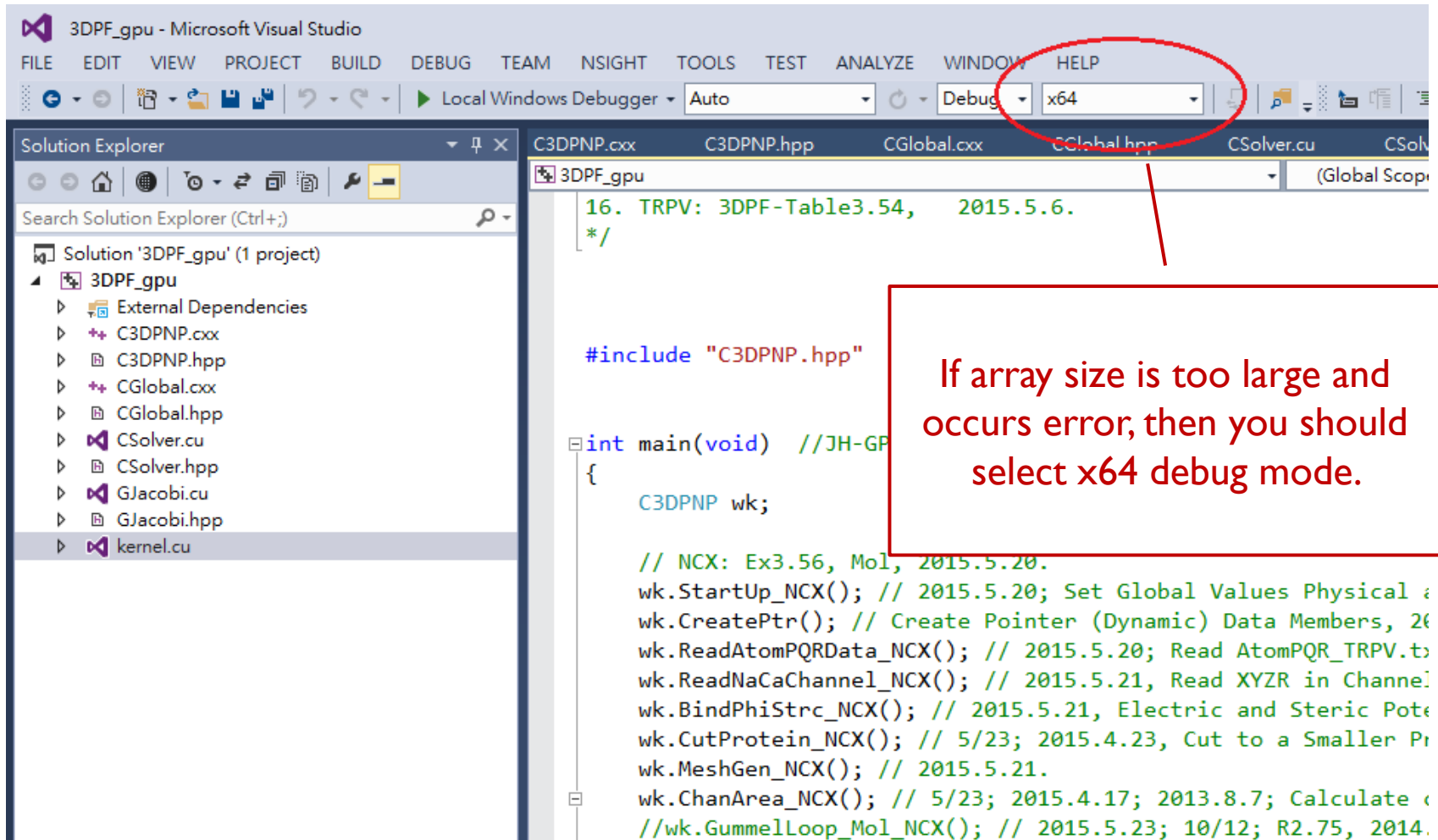
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#define BLOCK_NUM 1551 //1551, 12202
#define THREAD_NUM 512

__global__ void G_DiagMttxvec_v1( int N, double* g_invA0, double* g_A1, double* g_A2, double* g_A3, double* g_
double* g_b, double* g_xold, double* g_x, int XPts, int YPts, int XPtsYPts )
__global__ void G_DiagMttxvec_v2( int N, double* g_A, double* g_b, double* g_xold, double* g_x, int XPts, int
__global__ void G_DiagMttxvec_v3( int N, double* g_invA0, double* g_A1, double* g_A2, double* g_A3, double* g_
double* g_b, double* g_xold, double* g_x, int XPts, int YPts, int XPtsYPts )
__global__ void G_abserror( int N, double* g_xold, double* g_x, double* g_r, double* g_err );
__global__ void G_replacesol( int N, double* g_xold, double* g_x );

#endif
```

An example: 3DPF_gpu



3DPF_gpu - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM NSIGHT TOOLS TEST ANALYZE WINDOW HELP

Local Windows Debugger Auto Debug x64

Solution Explorer

Search Solution Explorer (Ctrl+;)

Solution '3DPF_gpu' (1 project)

- 3DPF_gpu
 - External Dependencies
 - C3DPNP.cxx
 - C3DPNP.hpp
 - CGlobal.cxx
 - CGlobal.hpp
 - CSolver.cu
 - CSolver.hpp
 - GJacobi.cu
 - GJacobi.hpp
 - kernel.cu

C3DPNP.cxx C3DPNP.hpp CGlobal.cxx CGlobal.hpp CSolver.cu CSol

3DPF_gpu (Global Scope)

```
16. TRPV: 3DPF-Table3.54, 2015.5.6.
*/

#include "C3DPNP.hpp"

int main(void) //JH-GP
{
    C3DPNP wk;

    // NCX: Ex3.56, Mol, 2015.5.20.
    wk.Startup_NCX(); // 2015.5.20; Set Global Values Physical &
    wk.CreatePtr(); // Create Pointer (Dynamic) Data Members, 20
    wk.ReadAtomPQRData_NCX(); // 2015.5.20; Read AtomPQR_TRPV.t
    wk.ReadNaCaChannel_NCX(); // 2015.5.21, Read XYZR in Channe
    wk.BindPhiStrc_NCX(); // 2015.5.21, Electric and Steric Pote
    wk.CutProtein_NCX(); // 5/23; 2015.4.23, Cut to a Smaller Pr
    wk.MeshGen_NCX(); // 2015.5.21.
    wk.ChanArea_NCX(); // 5/23; 2015.4.17; 2013.8.7; Calculate c
    //wk.GummelLoop_Mol_NCX(); // 2015.5.23; 10/12; R2.75, 2014.
```

If array size is too large and occurs error, then you should select x64 debug mode.

Reference

- ▶ NVIDIA [CUDA Installation Guide for Microsoft Windows](#)
- ▶ Microsoft Visual Studio 2013 語言套件
<https://www.microsoft.com/zh-tw/download/details.aspx?id=40783>

