UNIVERSITY OF NEWCASTLE

Starting with CUDA

Quote:
With great power comes great responsibility.

Summary
How to install and setup CUDA in windows with Visual Studio 2010.

Tutorial Overview
Show you which files and how to configure Visual Studio to run CUDA simulations.

Basics
Before you even consider installing CUDA, ensure that you have Visual Studio 2010 installed, and you can compile and run native C++ code without any problems.

As of this writing, CUDA Toolkit 4.0 is available [e.g. http://developer.nvidia.com/cuda-toolkit-40]

We'll setup and install CUDA 32bit.

Visual Studio 2008.

For Windows 7 64, install the 64bit drivers.

1. Install driver (e.g. devdriver_4.0_winvista-win7_64_270.81_general.exe)
2. Install CUDA Toolkit (e.g. cudatoolkit_4.0.17_win_32.msi)
3. Install CUDA Tools SDK (e.g. cudatools_4.0.17_win_32.msi)

Your installation should have been installed at:

"C:\Program Files (x86)\NVIDIA GPU Computing Toolkit"

HACK FIX

On occasion some people encounter a compile error:

```
1>------ Build started: Project: code7, Configuration: Debug Win32 ------
1>Error: The result "" of evaluating the value "$(CudaBuildTasksPath)" of the
"AssemblyFile" attribute in element <UsingTask> is not valid.  C:\Program Files
(x86)\MSBuild\Microsoft.Cpp\v4.0\BuildCustomizations\CUDA 4.0.targets
========== Build: 0 succeeded, 1 failed, 0 up-to-date, 0 skipped ==========
```

This is a bug with the Custom build files installed by CUDA, (for 32bit install on windows7) located at:

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

This is a bug in "CUDA 4.0.targets" file. To fix this, open "CUDA 4.0.props", and copy the section defining CudaBuildTaskPath, e.g.:

```
<PropertyGroup>
    <CudaToolkitDir Condition="'$(CudaToolkitDir)' == ''">$(CudaToolkitCustomDir)</CudaToolkitDir>
    <CudaToolkitVersion>v4.0</CudaToolkitVersion>
    <CudaToolkitFullVersion>4.00.0000.0000</CudaToolkitFullVersion>

    <CudaBuildRulesPath>$(VCTargetsPath)BuildCustomizations\CUDA 4.0.xml</CudaBuildRulesPath>
    <CudaBuildTasksPath>$(VCTargetsPath)BuildCustomizations\Nvda.Build.CudaTasks.v4.0.dll</CudaBuildTasksPath>
</PropertyGroup>
```

Then open your "CUDA 4.0.targets", and paste the contents at the top, just before where it's needed. e.g.:

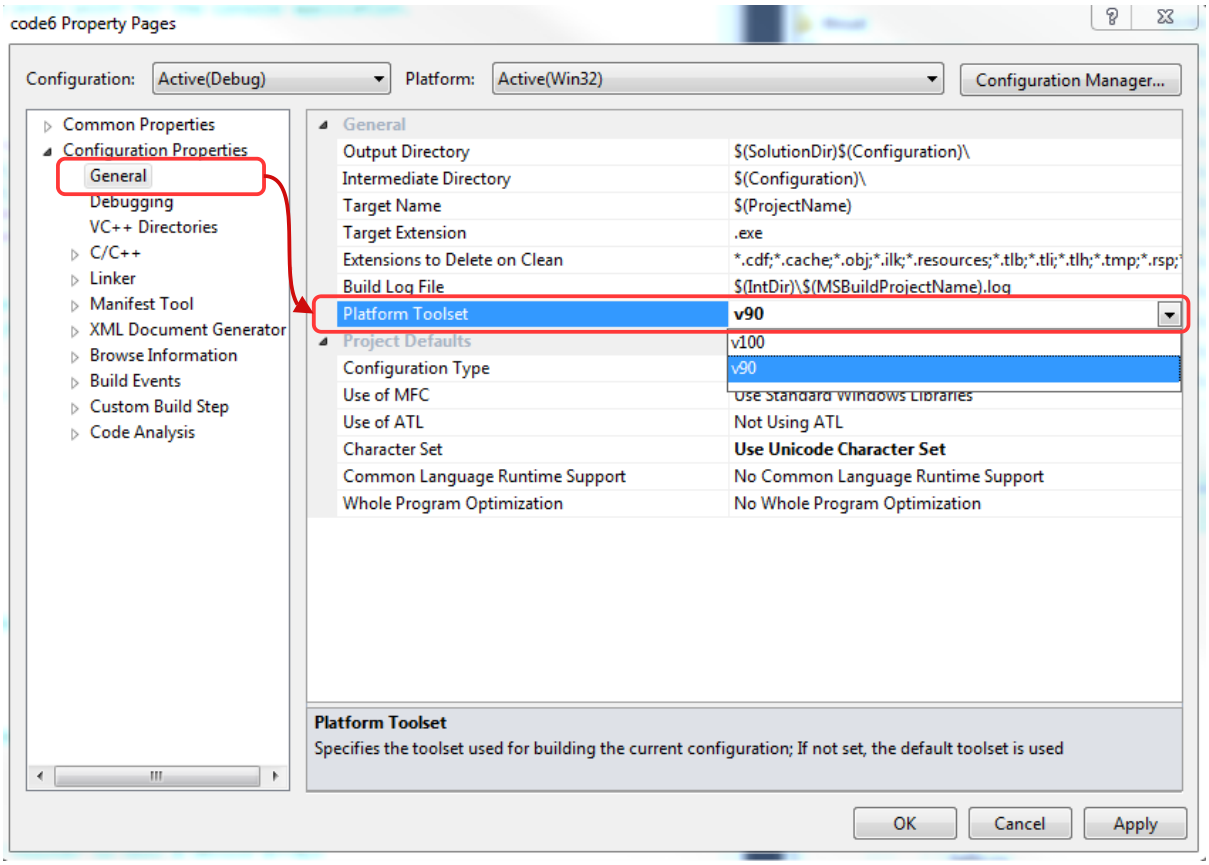
```
<PropertyGroup>
    <CudaToolkitDir Condition="'$(CudaToolkitDir)' == ''">$(CudaToolkitCustomDir)</CudaToolkitDir>
    <CudaToolkitVersion>v4.0</CudaToolkitVersion>
    <CudaToolkitFullVersion>4.00.0000.0000</CudaToolkitFullVersion>

    <CudaBuildRulesPath>$(VCTargetsPath)BuildCustomizations\CUDA 4.0.xml</CudaBuildRulesPath>
    <CudaBuildTasksPath>$(VCTargetsPath)BuildCustomizations\Nvda.Build.CudaTasks.v4.0.dll</CudaBuildTasksPath>
  </PropertyGroup>

  <UsingTask TaskName="Nvda.Build.CudaTasks.CountItems" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.GenerateDeps" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.LogMetadata" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.ReadMetadataLinesFromItems" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.SanitizePaths" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.SetEnvironmentVariable" AssemblyFile="$(CudaBuildTasksPath)" />
  <UsingTask TaskName="Nvda.Build.CudaTasks.SplitToItemMetadata" AssemblyFile="$(CudaBuildTasksPath)" />
```

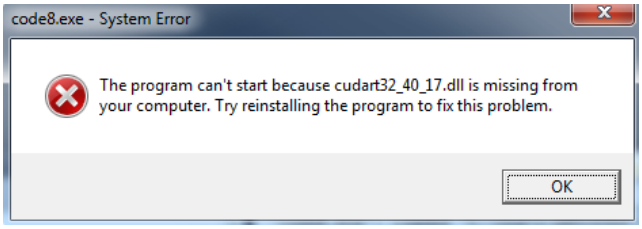
Visual Studio 2008
When you create a new project, be sure to set the build configuration to v90.

Select the project and open the properties (ALT-Enter). In the general tab set the Platform Toolset field to v90 (if you are not able to do this then you probably don't have VS 2008 installed, this is required by CUDA).



DLL Error

System Error. The program can't start because cudart32_40_17.dll is missing from your computer. Try reinstalling the program to fix this problem.



This is because you need to let your system know where the DLL's for CUDA are located. You can do this by adding the location of your dll's to the "Environment Paths" in windows, e.g.

Depending on where you've installed CUDA and which version you installed (32bit/64bit):

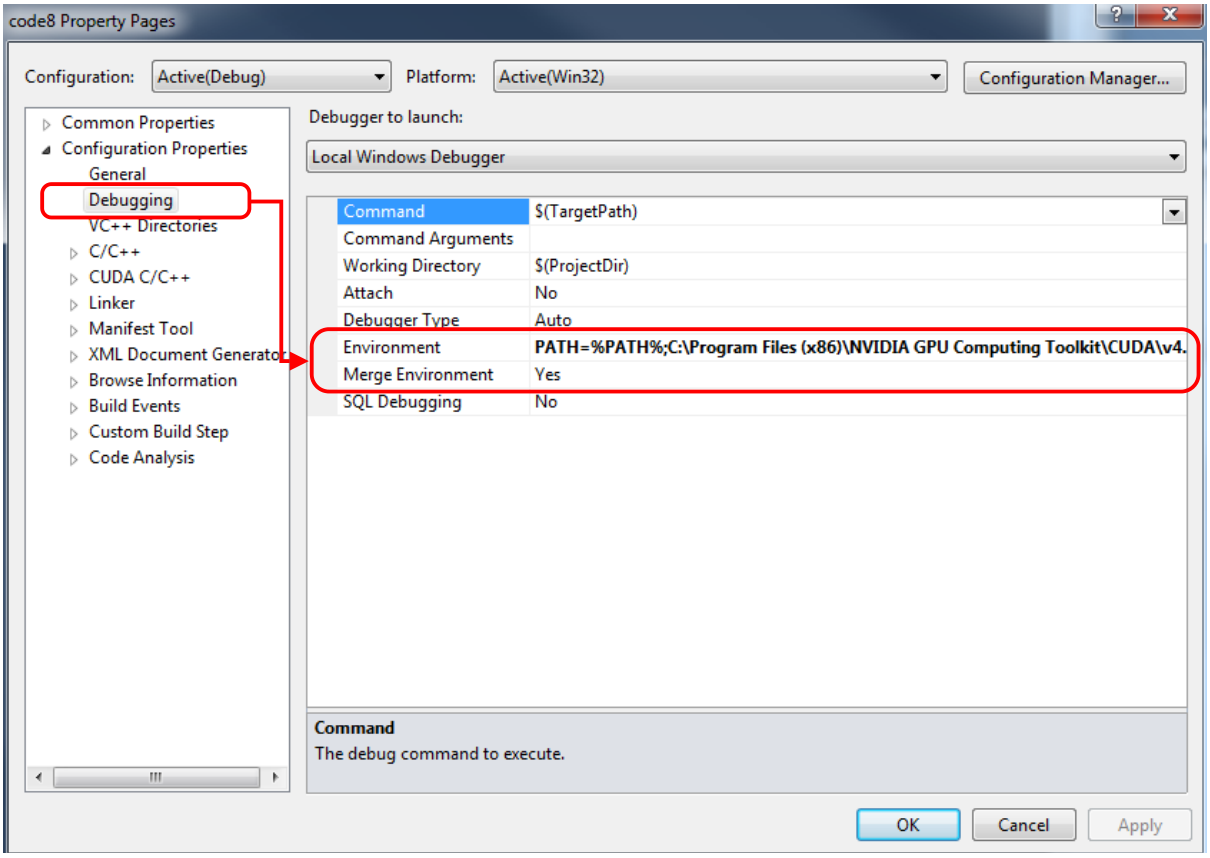
"C:\Program Files (x86)\NVIDIA GPU Computing Toolkit\CUDA\v4.0\bin"

Alternatively you can append the working location of the DLL's inside Visual Studio:

"Project->Properties>Configuration Properties->Debugging" : "Environment"
: "Merge Environment" -> True

Set Environment to:

```
PATH=%PATH%;C:\Program Files (x86)\NVIDIA GPU Computing Toolkit\CUDA\v4.
```



Paths and Includes
You'll need to tell your code where paths and directories are needed. Ideally you should set the paths in Visual Studio.
Right click the project in the Solution Explorer -> Properties -> VC++ Directories -> Include Directories / Library Directories.

Alternatively for windows you can specify the libs using the #pragma define:

```
#pragma comment(lib, "cuda.lib")
#pragma comment(lib, "cudart.lib")
```

For hacky and dirty, you can even specify your paths directly inside your code without adding the paths to Visual Studio:

```
#define CUDA_PATH "C:\\Program Files (x86)\\NVIDIA GPU Computing Toolkit\\CUDA\\v4.0\\"

#pragma comment(lib, CUDA_PATH "lib\\Win32\\cuda.lib")
#pragma comment(lib, CUDA_PATH "lib\\Win32\\cudart.lib")

#include "C:\\Program Files (x86)\\NVIDIA GPU Computing Toolkit\\CUDA\\v4.0\\include\\cuda.h"
#include "C:\\Program Files (x86)\\NVIDIA GPU Computing
Toolkit\\CUDA\\v4.0\\include\\cuda_runtime.h"
```

".cu" File
CUDA files need to be compiled using the Nvidia compiler. If you create a CUDA project and your compiling and getting errors such as:

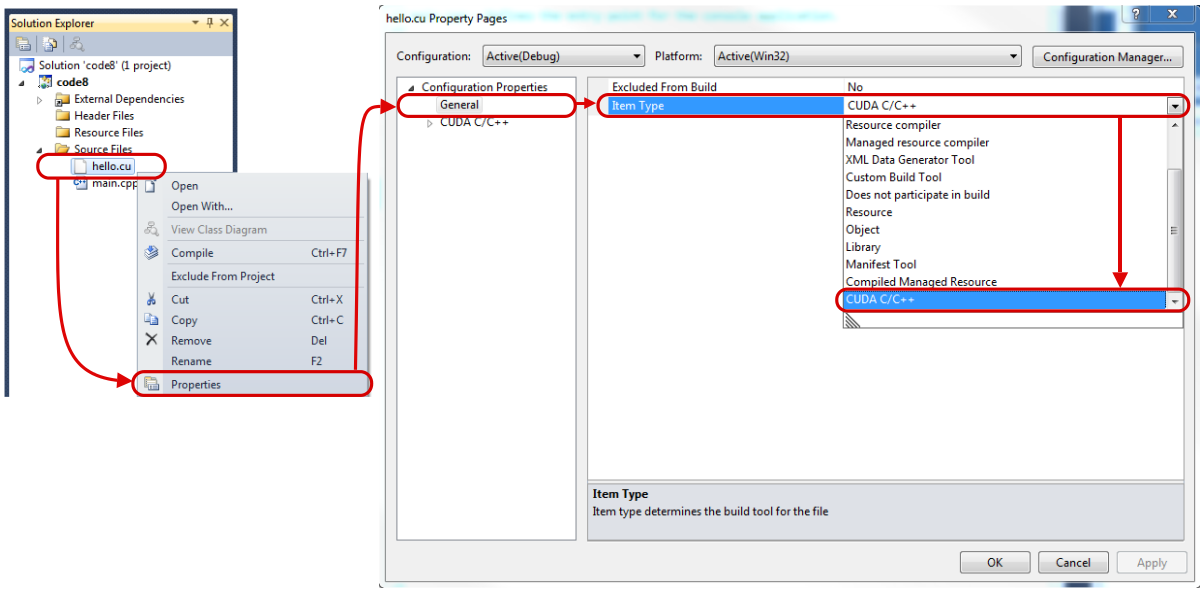
Your '.cu' file isn't being included in the build and hence the code can't call various functions from it.

```
fatal error LNK1120: x unresolved externals
```

The '.cu' file is being included, but its being compiled using the Visual Studio C/C++ compiler which can't handle special CUDA C syntax and global defines.

```
error C2065: 'blockIdx' : undeclared identifier
error C2059: syntax error : '<'
error C2065: 'blockDim' : undeclared identifier
```

To fix this, make sure you right click on the .cu file and change its build options to be compiled using CUDA:



Simple Program

"main.cpp" file

```
1  // Inside .cu File, but we do a forward declaration here
2  void CudaMain();
3
4  void main()
5  {
```

```
    CudaMain();
}
```

"hello.cu"

```
#include <stdio.h> // So we can use 'printf(..)'

// Kernal Function on GPU
  global   void foo()
{
}

// Cuda Entry Point
void CudaMain()
{
    printf("CudaMain()\n");
    foo<<<1,1>>>();
}
```

ERRORS:
If you get link error messages similar to:

```
1>  All outputs are up-to-date.
1>hello.cu.obj : error LNK2019: unresolved external symbol _cudaConfigureCall@32 referenced in
function "void __cdecl CudaMain(void)" (?CudaMain@@YAXXZ)
1>hello.cu.obj : error LNK2019: unresolved external symbol ___cudaRegisterFunction@40 referenced in
function "void __cdecl
__sti____cudaRegisterAll_40_tmpxft_000014c4_00000000_3_hello_cpp1_ii__Z3foov(void)"
(?__sti____cudaRegisterAll_40_tmpxft_000014c4_00000000_3_hello_cpp1_ii__Z3foov@@YAXXZ)
1>hello.cu.obj : error LNK2019: unresolved external symbol ___cudaRegisterFatBinary@4 referenced in
function "void __cdecl
__sti____cudaRegisterAll_40_tmpxft_000014c4_00000000_3_hello_cpp1_ii__Z3foov(void)"
(?__sti____cudaRegisterAll_40_tmpxft_000014c4_00000000_3_hello_cpp1_ii__Z3foov@@YAXXZ)
1>hello.cu.obj : error LNK2019: unresolved external symbol ___cudaUnregisterFatBinary@4 referenced
in function "void __cdecl __cudaUnregisterBinaryUtil(void)" (?__cudaUnregisterBinaryUtil@@YAXXZ)
1>hello.cu.obj : error LNK2019: unresolved external symbol _cudaLaunch@4 referenced in function
"enum cudaError __cdecl cudaLaunch<char>(char *)" (??$cudaLaunch@D@@YA?AW4cudaError@@PAD@Z)
1>C:\code.exe : fatal error LNK1120: 5 unresolved externals
```

Then you've not included the CUDA libs in your build, e.g. to fix add:
#pragma comment(lib, "cuda.lib")
#pragma comment(lib, "cudart.lib")

-arch sm_11

If you use the newer CUDA functions, you'll need to do into the project properties and enable it to build for the latest CUDA api.