CUDA Tutorial 01 - Windows

Getting up and running with CUDA and Visual Studio 2010 under Windows 7

Martin A. Kruse - June 2013

This document is part of a series of tutorials intended for the people at the extraterrestrial physics research group at the Christian Albrechts Universität zu Kiel. Everyone else is invited to use and distribute (as is) these documents, however the computers mentioned here are not available to the public.

The full series of tutorials can be found online at http://www.ieap.uni-kiel.de/et/people/kruse/, though access to this site later than 2026 might be impossible due to thermonuclear war.

Suggestions and corrections are very welcome at kruse@physik.uni-kiel.de.

This tutorial is meant to get you up and running with the CUDA computing platform utilizing Microsoft Visual Studio under Windows. Nothing useful will be computed, but the steps necessary to start any meaningful project are explained in detail.

This tutorial has been tested with Windows 7 64bit and Microsoft Visual Studio (MSVS) 2010 Professional, because the optional NSight Debugger only works with MSVS 2010 and 2008 releases. However, CUDA functionality should be possible with MSVS 2008 or newer or even the free Microsoft Visual C++ and the procedure is similar to the one described here.

1 Download and install the following ressources in that order:

1. Microsoft Visual Studio 2010 Professional (or similar)
   consult Windows Update now and get MSVS up-to-date.
2. CUDA Toolkit 5.0
   https://developer.nvidia.com/cuda-downloads
   - do not install drivers for the GPU device, that is to be done in the next step
   - Sample package is not required, but recommended because of tools which can help during development
   - use the default installation path, otherwise there might be configuration problems later on
3. (optional) NSight Visual Studio Edition (free registration required)
   https://developer.nvidia.com/rdp/nsight-visual-studio-edition-downloads
4. Install the latest drivers for your CUDA device
   http://www.geforce.com/drivers

2 Setting up MSVS

In order to obtain syntax highlighting for CUDA, navigate to
`C:\ProgramData\NVIDIA Corporation\CUDA Samples\v5.0\doc\syntax_highlighting\visual_studio_8`
and append the contents of usertype.dat to the one in
`C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE.`
If there is none, simply copy the file from the CUDA folder in there.

3 Create your first CUDA project

1. Start MSVS (choose C++ enviroment if asked)
2. Open new blank project:

   File → New → Project... → Empty Project → Enter "Tutorial01W" as name → Hit "OK"
3. Activate CUDA build customization:

   Right-click on project "Tutorial01W" → Build Customizations... → Check CUDA 5.0 → Hit "OK"
4. Add host part of the program:

   Right-click on "Source Files" → Add → New Item → C++ File (.cpp) → Enter "main.cpp" as name → Hit "Add"

   Enter or copy-and-paste the following source code

```
#include <stdio.h>

extern void cuda_doStuff(void);

int main( int argc, const char* argv[] )
{
    cuda_doStuff();
}
```

5. Add CUDA part of the program:

   Right-click on “Source Files” → Add → under “Installed Templates”, choose NVIDIA→CUDA → CUDA C/C++ File → Enter “cuda_wrapper.cu” as name → Hit “Add”

   Enter or copy-and-paste the following source code

```
#include <cuda.h>
#include <cuda_runtime.h>
#include <stdio.h>

__global__ void someKernel(int N)
{
    int idx = blockIdx.x*blockDim.x + threadIdx.x;

    if (idx<N)
        printf("Hello␣from␣thread␣#␣%i␣(block␣#:␣%i)\n", idx, blockIdx.x);
}

extern void cuda_doStuff(void)
{
    int numberOfBlocks = 2;
    int threadsPerBlock = 5;
    int maxNumberOfThreads = 10;
    someKernel<<<numberOfBlocks, threadsPerBlock>>>(maxNumberOfThreads);
    cudaDeviceSynchronize();
}
```

6. Adjust the compute capability according to your device. See, e.g., http://en.wikipedia.org/wiki/CUDA for a detailed list of supported CUDA devices (for the GTX 480, choose compute capability 2.0, for GTX Titan choose 3.5, or below). This tutorial requires at least compute capability 2.0.

   Right-click on project “Tutorial01W” → Properties → CUDA C/C++ → Device → Enter compute_x.x, sm_x.x under “Code Generation” and adjust x.x accordingly

   While you are already there, change “Verbose PTXAS Output” to “Yes”. This will give some very useful information while compiling your kernels.

   Acknowledge by hitting “OK”.

7. Add Library Links

   Right-click project “Tutorial01W” → Properties → Linker → Input → Add cuda.lib and cudart.lib to “Additional Dependencies”

8. Compile the program by pressing F7

9. Run the program by opening a command window and navigating to the Debug folder of your project. Type “Tutorial01W” and hit enter (you might as well run it from inside MSVS, but that window vanishes very fast so you won’t see much). Congratulations, you have programmed and executed your first CUDA program.

Some notes: The printf-command is a host function, that has been added to the CUDA enviroment to help debug programs. Of course data has to be transfered back to the host in order to put it out in the console. It should therefore not be used in any purposeful context, as it probaly slows down computation time significantly