How to install OpenCL on Windows

Getting your Windows machine ready for OpenCL is rather straightforward. In short, you only need the latest drivers for your OpenCL device(s) and you're ready to go. Of course, you will need to add an OpenCL SDK in case you want to develop OpenCL applications but that's equally easy.

Before we start, a few notes:

- The steps described herein have been tested on Windows 8.1 only, but should also apply for Windows 7 and Windows 8. - We will not discuss how to write an actual OpenCL program or kernel, but focus on how to get everything installed and ready for OpenCL on a Windows machine. This is because writing efficient OpenCL kernels is almost entirely OS independent.

If you want to know more about OpenCL and you are looking for simple examples to get started, check the Tutorials section on this webpage.

Running an OpenCL application]

If you only need to run an OpenCL application without getting into development stuff then most probably everything already works. If OpenCL applications fail to launch, then you need to have a closer look to the drivers and hardware installed on your machine:

GPU Caps Viewer

- Check that you have a device that supports OpenCL. All graphics cards and CPUs from 2011 and later support OpenCL. If your computer is from 2010 or before, check this page. You can also find a list with OpenCL conformant products on Khronos webpage.] - Make sure your OpenCL device driver is up to date, especially if you're not using the latest and greatest hardware. With certain older devices OpenCL support wasn't initially included in the drivers.

Here is where you can download drivers manually:

- Intel has hidden them a bit, but you can find them here with support for OpenCL 2.0.] - AMD's GPU-drivers include the OpenCL-drivers for CPUs, APUs and GPUs, version 2.0.]

NVIDIA's GPU-drivers mention mostly CUDA, but the drivers for OpenCL 1.1 1.2 are there too.

In addition, it is always a good idea to check for any other special requirements that the OpenCL application may have. Look for device type and OpenCL version in particular. For example, the application may run only on OpenCL CPUs, or conversely, on OpenCL GPUs. Or it may require a certain OpenCL version that your device does not support.

A great tool that will allow you to retrieve the details for the OpenCL devices in your system is Caps Viewer.

Developing OpenCL applications]

Now it's time to put the pedal to the metal and start developing some proper OpenCL applications.

The basic steps would be the following:

- Make sure you have a machine which supports OpenCL, as described above. - Get the OpenCL headers and libraries included in the OpenCL SDK from your favourite vendor. - Start writing OpenCL code. That's the difficult part. - Tell the compiler where the OpenCL headers are located. - Tell the linker where to find the OpenCL .lib files. - Build the fabulous application. - Run and prepare to be awed in amazement.

Ok, so let's have a look into each of these.

OpenCL SDKs]

For OpenCL headers and libraries the main options you can choose from are:

- NVIDIA ? CUDA Toolkit. You can grab the OpenCL samples here.]

AMD ? AMD APP SDK. Also works with Intel's CPUs.

- Headers and OpenCL.lib are here: https://github.com/GPUOpen-LibrariesAndSDKs/OCL-SDK/releases] - Samples are here: https://github.com/OpenCL/AMD_APP_samples]

Math libraries are here: https://github.com/clMathLibraries - Intel ? the previous Intel SDK for OpenCL is now integrated into Intel's new tools, such as Intel INDE (which has a free starters edition) or Intel Media Server Studio. Grab any of these in order to have everything ready for building OpenCL code.]

As long as you pay attention to the OpenCL version and the OpenCL features supported by your device, you can use the OpenCL headers and libraries from any of these three vendors.

OpenCL headers]

Let's assume that we are developing a 64bit C/C++ application using Visual Studio 2013. To begin with, we need to check how many OpenCL platforms are available in the system:

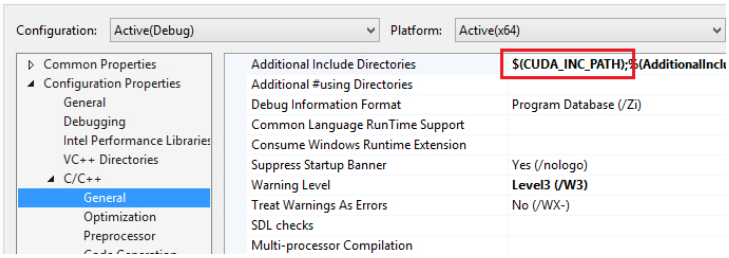
[raw]

```
#include<stdio.h>
#include<CL/cl.h>
int main(void)
{
   cl_int err;
   cl_uint numPlatforms;
   err = clGetPlatformIDs(0, NULL, &numPlatforms);
   if (CL_SUCCESS == err)
      printf("nDetected OpenCL platforms: %d", numPlatforms);
   else
      printf("nError calling clGetPlatformIDs. Error code: %d", err);
   return 0;
}
```

[/raw]

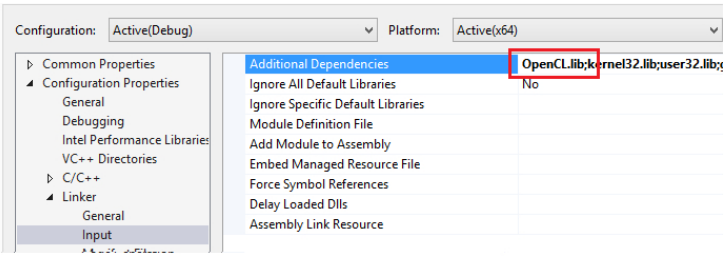
We need to specify where the OpenCL headers are located by adding the path to the OpenCL "CL" is in the same location as the other CUDA include files, that is, CUDA_INC_PATH. On a x64 Windows 8.1 machine with CUDA 6.5 the environment variable CUDA_INC_PATH is defined as ?C:Program FilesNVIDIA GPU Computing ToolkitCUDAv6.5include?

If you're using the AMD SDK, you need to replace "$(CUDA_INC_PATH)" with "$(AMDAPPSDKROOT)/include" or, for Intel SDK, with "$(INTELOCLSDKROOT)/include".

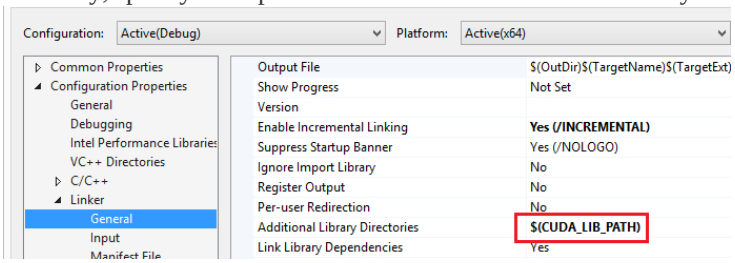


OpenCL libraries]

Similarly, we need to let the linker know about the OpenCL libraries. Firstly, add OpenCL.lib to the list of Additional Dependencies:



Secondly, specify the OpenCL.lib location in Additional Library Directories:



As in the case of the includes, If you're using the AMD SDK, replace "$(CUDA_LIB_PATH)" with "$(AMDAPPSDKROOT)/lib/x86_64" , or in the case of Intel with "$(INTELOCLSDKROOT)/lib/x64".

And you're good to go! The application should now build and run. Now, just how difficult was it? Happy OpenCL-coding on Windows!

If you have any question or suggestion, just leave a comment.