AMD **Accelerated** Parallel Processing TECHNOLOGY

AMD APP SDK v2.9.1

Getting Started

1 Overview

The AMD APP SDK is provided to the developer community to accelerate the programming in a heterogeneous environment by enabling AMD GPUs to work in concert with the system's x86 CPU cores. The SDK provides samples, documentation, and other materials to quickly get you started leveraging accelerated compute using OpenCL™, Bolt, OpenCV, C++ AMP for your C/C++ application, or Aparapi for your Java application.

This document provides instructions on using the AMD APP SDK. The necessary prerequisite installations, environment settings, build and execute instructions for the samples are provided.

Review the following quick links to the important sections:

2 APP SDK on Windows

2.1 Installation

The AMD APP SDK 2.9.1 installer is delivered as a self-extracting installer for 32-bit and 64-bit systems on Windows. For details on how to install the APP SDK on Windows, see the AMD APP SDK Installation Notes document. The default installation path is `C:\Users\<userName>\AMD APP SDK\<appSdkVersion>\`.

2.2 General Prerequisites

AMD APP SDK 2.9.1 is supported on the following Windows versions:

- Windows 8.1 (32-bit/64-bit)
- Windows 7 (32-bit/64-bit)

The AMD APP SDK includes sample applications for OpenCL, Bolt, C++ AMP, Aparapi and OpenCV-CL. To build and execute these samples, you must ensure that two sets of prerequisites are met: the common prerequisites that apply to all the samples, listed in this section; and the specific prerequisites required for the specific category of the samples, listed in the later sections of this document.

Before you build and execute the samples, ensure that you have installed the following:

- Microsoft Visual Studio 2012 redistributable package
  This package is required for running the pre-built binaries of the samples. Installing the AMD Catalyst Driver installs the Visual Studio 2012 redistributable by default.
- Microsoft Visual Studio version 2010 and/or higher
  This package is required for building and executing the samples.
- CMake version 2.8.0 or higher (optional)
  CMake is used for generating the AMD APP SDK sample make files or Visual Studio projects. In addition to Visual Studio solution/project files for samples, the AMD APP SDK includes CMake files. CMake supports creating `make` files across different platforms and generating project files across different IDEs including Visual Studio.
  For details on using CMake to generate make files or Visual Studio files for APP SDK samples, see Appendix C, “CMAKE”.

Note: The AMD APP SDK 2.9.1 package for Windows includes Visual Studio 2010 and 2012 projects. Visual Studio 2013 projects are not provided in the AMD APP SDK package. However, the CMake files of the samples can be used to generate VS 2013 project files.

2.3 OpenCL

2.3.1 Prerequisites

In addition to the common prerequisites mentioned in Section 2.2, “General Prerequisites”, to build and run OpenCL samples that use OpenGL and DirectX, you must install one of the following:

- Windows SDK 8.0 or above OR

2.3.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of OpenCL samples. To execute the samples, you must perform the following steps:

- Open a command prompt.
- Change the directory to the appropriate architecture directory (x86 or x86_64) within <<*APPSDKSamplesInstallPath*>>\samples\opencl\bin.
- Run the samples by typing the name of the executables.
  To review the command line arguments for samples, see the respective sample documents.

2.3.3 How to rebuild the samples

Building with Visual Studio Solution files –

The AMD APP SDK installation includes a master Visual Studio Solution file for OpenCL samples. This solution file contains Visual Studio projects of all the OpenCL samples. In the current version of APP SDK, master solution files for Microsoft Visual Studio 2010 (OpenCLSamplesVS10.sln) and Microsoft Visual Studio 2012(OpenCLSamplesVS12.sln) are provided. These files are located at <<*APPSDKSamplesInstallPath*>>\samples\opencl\.

To build a sample:

- Open the OpenCLSamplesVS10.sln file with Microsoft Visual Studio 2010 Professional Edition or the OpenCLSamplesVS12.sln file with Microsoft Visual Studio 2012 Professional Edition.
- To build all the projects, select Build > Build Solution.
  To build a specific project, select the project file in the Solutions Explorer and select Build to build the particular sample.

Building with Visual Studio Solution files by using the Intel Compiler (icl) –

The AMD APP SDK installation includes a master Visual Studio Solution file for OpenCL samples. This solution file contains Visual Studio projects of all the OpenCL samples. In the current version of APP SDK, master solution files for Microsoft Visual Studio 2010 (OpenCLSamplesVS10.sln) and Microsoft Visual Studio 2012(OpenCLSamplesVS12.sln) are provided. These files are located at <<*APPSDKSamplesInstallPath*>>\samples\opencl\.

To build a sample:

- Open the OpenCLSamplesVS10.sln file with Microsoft Visual Studio 2010 Professional Edition or the OpenCLSamplesVS12.sln file with Microsoft Visual Studio 2012 Professional Edition.
- Right-click on a project file, and select Properties.
- Under Configuration Properties | General, change the Platform Toolset item to Intell C++ Compiler, and click OK.
- To build the sample, right-click on the project file, and select Build.

2.4 BOLT

2.4.1 Prerequisites

In addition to the common prerequisites mentioned in Section 2.2, "General Prerequisites", to build and run the BOLT samples you must ensure the following installations and environment variable settings:

- Microsoft Visual Studio 2012 or higher for Bolt C++AMP samples and Microsoft Visual Studio 2010 or higher for Bolt OpenCL samples
- The TBB library: http://threadingbuildingblocks.org/download.
  - Required for running the Bolt samples using the multi-core CPU path.
  - BOLT 1.2 has been tested with the TBB release 4.2 Update 4.
- Download and install the Bolt 1.2 library: http://developer.amd.com/tools-andsdks/opencl-zone/opencl-libraries/bolt-c-template-library/.
  Bolt can also be built from the *github* sources: https://github.com/HSA-Libraries/Bolt/releases/tag/v1.2GA.

Environment Variables –

Set the following environment variables:

- Set the `TBB_ROOT` environmental variable to the root directory of the installed TBB binaries.
- Set the `BOLTLIB_DIR` environmental variable to the root directory to which Bolt is extracted. For example, if VS 2012 Bolt 1.2 is downloaded, the path will be: `<Bolt Install Path>\Bolt-1.1-VS20123\Bolt-1.1-VS2012\`.
- Append the `PATH` environment variable with the directory containing all the TBB binaries. For example, on a 64-bit machine with VS 2012, this path will be: `%TBB_ROOT%\bin\intel64\vc11\`.

2.4.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of BOLT samples. To execute the samples, you must perform the following steps:

- Open a command prompt.
- Change the directory to the appropriate directory (x86 or x86_64) within `<<APPSDKSamplesInstallPath>>\samples\bolt\bin`.
- Run the samples by typing the name of their executables.
  To review the command line arguments for samples, see the respective sample documents.

2.4.3 How to rebuild the samples

The AMD APP SDK installation includes includes a master Visual Studio Solution file for Bolt samples. This solution file contains Visual Studio projects of all the Bolt samples. In the current version of APP SDK, master solution files for Microsoft Visual Studio 2010 (BoltSamplesVS10.sln) and Microsoft Visual Studio 2012 (BoltSamplesVS12.sln) are provided. These files are located at `$<APPSDKSamplesInstallPath>\samples\bolt\`.

To build a sample:

- Open the `BoltSamplesVS10.sln` or `BoltSamplesVS12.sln` file.
- Select the appropriate build configuration, `Debug` or `Release`.
- To build TBB-enabled Bolt samples, select the corresponding build configuration, `Debug_TBB` or `Release_TBB`.
  To build all the projects, select Build > Build Solution.
  To build a specific project. select the project file in the Solutions Explorer and select Build to build the sample.

2.5 C++ AMP

2.5.1 Prerequisites

C++ AMP is supported only on Microsoft Visual Studio 2012 and higher versions. Also, C++ AMP samples do not work on Linux. Therefore, to build and run C++AMP samples, install Microsoft Visual Studio 2012 or higher.

2.5.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of the C++AMP samples. To execute the samples, you must perform the following steps:

- Open a command prompt.
- Change the directory to the appropriate architecture directory (x86 or x86_64) within `<<APPSDKSamplesInstallPath>>\samples\C++Amp\bin`.
- Run the samples by typing the name of the executables.
  To review the command line arguments for samples, see the respective sample documents.

2.5.3 How to rebuild the samples

The AMD APP SDK installation includes a master Visual Studio Solution file for the C++AMP samples. This solution file contains Visual Studio projects of all the C++AMP samples. In the current version of APP SDK, master solution files for Microsoft Visual Studio 2012 (`C++AmpSamplesVS12.sln`) are provided. These files are located at `$<APPSDKSamplesInstallPath>\samples\C++Amp\`.

To build a sample:

- Open the `C++AmpSamplesVS12.sln` file.
- Select the appropriate build configuration, `Debug` or `Release`.
- To build all the projects, select Build > Build Solution.
  To build a specific project, select the project file in the Solutions Explorer and select Build to build the sample.

2.6 Aparapi

The samples for Aparapi are different from the samples for the other platforms in that all the Aparapi samples use Java. Eclipse can be used to build and execute Aparapi samples.

2.6.1 Prerequisites

To build and run the Aparapi samples, you must ensure that the following prerequisites are met:

- Install Java JDK 7: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.
- Install the ANT build tool: https://ant.apache.org/bindownload.cgi.
- Install the Aparapi library. Aparapi samples work with the latest Aparapi trunk source in the link http://code.google.com/p/aparapi/source/browse/#svn%2Ftrunk. Source files from this link must be fetched and built. The procedure to compile the sources is provided in the wiki pages of Aparapi:
  - https://code.google.com/p/aparapi/wiki/DevelopersGuideWindows

To build and run the Aparapi examples, you must configure the following environment variables:

- Set `JAVA_HOME` to the directory containing JRE/JDK.
  For example, specify: `set JAVA_HOME=C:\Program Files\Java\jdk1.7.0_55\`.
- Set `ANT_HOME` to the directory containing ANT, version 1.8 or above.
  For example, if ANT is installed under `C:\ANT`, specify: `set ANT_HOME=C:\Ant\apache-ant-1.9.3-bin\apache-ant-1.9.3\`.
- Set `LIBAPARAPI` to the directory in which the Aparapi binaries (`aparapi.jar`) are created after compiling.
  For example, specify: `LIBAPARAPI = C:\Aparapi\`.
- Append the `PATH` environment variable with the directory containing the Java, ANT, and Aparapi binaries, i.e. `%JAVA_HOME%\bin`, `%ANT_HOME%\bin`, and `%LIBAPARAPI%` (needed if running from Eclipse).

2.6.2 How to run the pre-built samples

The prebuilt jar files for the Aparapi samples are located in the respective sample directory. For example, the Mandel prebuilt jar file for the Mandel sample is located under:
`$<APPSDKSamplesInstallPath>\samples\aparapi\examples\Mandel\`.

To execute the sample, run the appropriate `<example>.bat` from the command line.
`<example>.bat -h` lists the command-line options provided by the sample.

2.6.3 How to rebuild the samples

In order to build all the samples, run the `build.bat` batch file under
`$<APPSDKSamplesInstallPath>\samples\aparapi`.

To build individual samples, go to the respective sample directory and type `ant`.
For example, to compile the Mandel sample, run:
`$<APPSDKSamplesInstallPath>\samples\aparapi\examples\Mandel> ant`

Using Eclipse –

The APP SDK Aparapi samples also have Eclipse project files to allow building and executing using the Eclipse IDE. AMD APP SDK 2.9.1 has been verified to work with Eclipse Standard 4.3.2 version (https://www.eclipse.org/downloads/packages/eclipse-standard-432/keplersr2).

In addition to the general prerequisites mentioned earlier for Aparapi samples, define a Classpath variable, `LIBAPARAPI`, in the Eclipse IDE, by performing the following steps:

1. From the Eclipse IDE menu bar, select Window > Preferences.
2. Select Java > Build Path > Classpath Variables.
3. Click the **NEW > define a new LIBAPARAPI variable** button and point it to your local `Aparapi.jar` folder.
4. Close the Eclipse IDE.
5. To build the project, open the Eclipse IDE, import the `AparapiUtil` project, and build the project.
   Similarly, you can import all the Aparapi sample projects and build and execute them.

2.7 OpenCV

2.7.1 Prerequisites

The AMD APP SDK 2.9.1 OpenCV-CL samples work with OpenCV 2.4.9. Download the OpenCV 2.4.9 binaries for Windows from: http://opencv.org/.

The Windows download includes the pre-built OpenCV binaries. However, you can also create binaries by downloading the source files and building them, as described in Appendix D, "Building OpenCV from sources". For answers to frequently asked questions on OpenCV, see the AMD APP SDK FAQ.

Environment Variables –

Before building and running the OpenCV-CL samples, you must set the following environment variables:

- Create and set the environmental variable, `OPENCV_DIR`, to the root directory containing the OpenCV include and lib files that are extracted from the downloaded OpenCV package.
- Create and set the environmental variable, `OCVCL_VER`, to the OpenCV version used for APP SDK 2.9.1 release, that is, set `OCVCL_VER = 249`.
- Append the `PATH` environment variable with the directory containing all the OpenCV .dll files. For example, for a 64-bit machine with VS 2012, this path will be `%OPENCV_DIR%\x64\vc11\bin`.

OpenNI Libraries –

The GestureRecognition APP SDK OpenCV-CL sample makes use of OpenNI libraries to extract video frames. OpenNI framework is an open source SDK used for the development of 3D sensing middleware libraries and applications. The OpenNI SDK can be downloaded from http://structure.io/openni.

You must set the following environment variables:

1. For 32-bit builds, add `OPENNI2_REDIST` to the `PATH` environment variable.
   For 64-bit builds, add `OPENNI2_REDIST64` to the `PATH` environment variable.

2. Header and library paths will be added by the OpenNI Windows installer. If those paths are missing from the system, then the user must set the following environment variables:
    i. For 32-bit platforms:
    Set `OPENNI2_INCLUDE` to `<<OPENNI-INSTALL_PATH>>\Include`
    Set `OPENNI2_LIB` to `<<OPENNI-INSTALL_PATH>>\Redist`
    Add `OPENNI2_LIB` to the `PATH` environment variable
    ii. For 64-bit platforms:
    Set `OPENNI2_INCLUDE64` to `<<OPENNI-INSTALL_PATH>>\Include`
    Set `OPENNI2_LIB64` to `<<OPENNI-INSTALL_PATH>>\Redist`
    Add `OPENNI2_LIB64` to the `PATH` environment variable

The GestureRecognition sample currently works on only Windows platforms.

2.7.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of the OpenCV-CL samples. To execute the samples, you must perform the following steps:

- Open a command prompt.
- Change the directory to the appropriate architecture directory (x86 or x86_64) within `<<APPSDKSamplesInstallPath>>\samples\opencv\bin`.
- Run the samples by typing the name of the executables.
  To review the command line arguments for samples, see the respective sample documents.

2.7.3 How to rebuild the samples

The AMD APP SDK installation includes a master Visual Studio Solution file for the OpenCV-CL samples. This solution file contains Visual Studio projects of all the OpenCV-CL samples. In the current version of APP SDK, master solution files for Microsoft Visual Studio 2010 (`OpenCVSamplesVS10.sln`) and Microsoft Visual Studio 2012 (`OpenCVSamplesVS12.sln`) are provided. These files are located at `$<APPSDKSamplesInstallPath>\samples\opencv\`.

To build a sample:

- Open the `OpenCVSamplesVS10.sln` file or the `OpenCVSamplesVS12.sln` file.
- Select the appropriate build configuration, `Debug` or `Release`.
- To build TBB-enabled Bolt samples, select the corresponding build configuration, `Debug_TBB` or `Release_TBB`.
- To build all the projects, select Build > Build Solution.
  To build a specific project, select the project file in the Solutions Explorer and select Build to build the sample.

3 APP SDK on Linux

3.1 Installation

The AMD APP SDK 2.9.1 installer is delivered as a self-extracting installer for 32-bit and 64-bit systems on Linux. For details on how to install APP SDK on Linux, see the AMD APP SDK Installation Notes document.

3.2 General prerequisites

The AMD APP SDK 2.9.1 is supported on the following Linux flavors:

- openSUSE™ 13.1 (32-bit/64-bit)
- Ubuntu® 14.04 (32-bit/64-bit)
- Red Hat® Enterprise Linux® 6.*x* (32-bit/64-bit)

The AMD APP SDK Linux package includes sample applications for OpenCL, Bolt, Aparapi and OpenCV-CL. To build and execute these samples, you must ensure that two sets of prerequisites are met: the common prerequisites that apply to all the samples, listed in this section; and the specific prerequisites required for the specific category of the samples, listed in the later sections of this document.

Before you build and execute the samples, ensure that you have installed the following:

- `gcc`
  - The AMD APP SDK samples have been compiled with `gcc 4.7.3`
- CMake version 2.8.0 or higher
  - For details on using CMake to generate make files for the AMD APP SDK samples, see Appendix C, “CMAKE”.

3.3 OpenCL

3.3.1 Prerequisites

In addition to the common prerequisites mentioned in Section 3.2, “General prerequisites”, to build and run the OpenCL samples you must ensure the following installation:

- OpenGL package `freeglut3-dev`
  - Required to run AMD APP SDK samples that use OpenGL library
  - Ensure that the AMD Catalyst Driver is installed in order to run samples using OpenGL
  - Ensure that the `libGL.so.1` file is linked to `fglrx-libGL.so.1.2`, which is found in:
    Ubuntu: `/usr/lib/fglrx/` and `/usr/lib32/fglrx/`
    RHEL: `/usr/lib64/fglrx/`.
    The sample execution fails when linked to:
    Ubuntu: `/usr/lib/x86_64-linuxgnu/mesa/libGL.so.1`, `/usr/lib/i386-`

linux-nu/mesa/libGL.so.1
RHEL: /usr/lib/libGL.so.1, /usr/lib64/libGL.so.1.

3.3.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of OpenCL samples. To execute the samples, you must perform the following steps:

- Open a terminal window.
- Change the directory to the appropriate architecture directory (x86 or x86_64) within *<<APPSDKSamplesInstallPath>>*/samples/opencl/bin.
- Run the samples by typing the name of the executables. You may need to prepend the executable name with ./.
  To review the command line arguments for samples, see the respective sample documents.

Note: The prebuilt samples on Linux have been compiled with GCC 4.7.3.

3.3.3 How to rebuild the samples

To compile the AMD APP SDK OpenCL samples, the CMake build tool is required. To build all the OpenCL samples, run the following commands:

```
$> cmake -G "Unix Makefiles"
"<APPSDKSamplesInstallPath>/samples/opencl/"
$> make
```

To build individual samples, run CMake from the respective sample source directory.

For more details on using CMake to generate make files for the AMD APP SDK samples, see Appendix C, “CMAKE”.

3.4 BOLT

3.4.1 Prerequisites

In addition to the common prerequisites mentioned in Section 3.2, “General prerequisites”, to build and run the BOLT samples you must ensure the following installations and environment variable settings:

- The TBB library: http://threadingbuildingblocks.org/download.
  - Required for running the Bolt samples using the multi-core CPU path.
  - BOLT 1.2 has been tested with the TBB release 4.2 Update 4.
- Download and install the Bolt 1.2 library for Linux from: http://developer.amd.com/tools-andsdks/opencl-zone/opencl-libraries/bolt-c-template-library/.
  Bolt can also be built from the *github* sources: https://github.com/HSA-Libraries/Bolt/releases/tag/v1.2GA.

Environment Variables –

Set the following environment variables:

- Set and export the `TBB_ROOT` environmental variable to the root directory of the installed TBB binaries.
- Set and export the `BOLTLIB_DIR` environmental variable to the root directory to which Bolt is extracted.
- Append the `LD_LIBRARY_PATH` environment variable with the directory containing all the TBB binaries. For example, on a 64-bit machine, this path can be: `/home/<user>/downloads/TBB/tbb42_xxxxyyzzoss/lib/intel64/gcc<majorVersion.minorVersion>`.

3.4.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of BOLT samples. To execute the samples, you must perform the following steps:

- Open a terminal window.
- Change the directory to the appropriate directory (x86 or x86_64) within `<<APPSDKSamplesInstallPath>>/samples/bolt/bin`.
- Run the samples by typing the name of their executables. You may need to prepend the name of the executable with `./`.
  To review the command line arguments for samples, see the respective sample documents.

3.4.3 How to rebuild the samples

To compile the AMD APP SDK Bolt samples, the CMake build tool is required. To build all the OpenCL samples, run the following commands:

```
$> cmake -G "Unix Makefiles" "<APPSDKSamplesInstallPath>/samples/bolt/"
$> make
```

To build individual samples, run CMake from the respective sample source directory.

To build a Bolt sample with the TBB multicore option enabled, use the `ENABLE_TBB` flag.

For example:

```
$> cmake -G "Unix Makefiles" -DENABLE_TBB=ON
"<APPSDKSamplesInstallPath>samples/bolt/BoltSort/"
$> make
```

For more details on using CMake to generate make files for the AMD APP SDK samples, see Appendix C, “CMAKE”.

3.5 Aparapi

The samples for Aparapi are different from the samples for the other platforms in that all the Aparapi samples use Java. Eclipse can be used to build and execute Aparapi samples.

3.5.1 Prerequisites

In addition to the common prerequisites mentioned in Section 3.2, “General prerequisites”, to build and run the Aparapi samples you must ensure the following installations and environment variable settings:

Installations –

- Install the Java JDK: http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html.
- Install the ANT build tool: https://ant.apache.org/bindownload.cgi.
- Install the Aparapi library. The AMD APP SDK 2.9.1 Aparapi samples work with the latest Aparapi trunk source in the link http://code.google.com/p/aparapi/source/browse/#svn%2Ftrunk. Source files from this link must be fetched and built. The procedure to compile the sources is provided in the wiki pages of Aparapi:
  - https://code.google.com/p/aparapi/wiki/DevelopersGuideLinux

Environment Variables –

- Set and export `JAVA_HOME` to the directory containing JRE/JDK.
  For example, `JAVA_HOME=/home/<user>/downloads/Java/jdk1.7.0_55`.
- Set `ANT_HOME` to the directory containing ANT, version 1.8 or above.
  For example, if ANT is installed under `/home/<user>/downloads/ANT`, set `ANT_HOME=/home/<user>/downloads/Ant/apache-ant-1.9.3`.
- Set and export `LIBAPARAPI` to the directory in which the Aparapi binaries (`aparapi.jar`) are created after compiling.
  For example, specify: `LIBAPARAPI = /home/<user>/downloads/Aparapi/`.
- Append and export the `PATH` environment variable to the directory containing the Java, ANT, and Aparapi binaries, i.e. `$JAVA_HOME/bin` and `$ANT_HOME/bin`.
- If running from Eclipse, set `LD_LIBRARY_PATH` to the `$LIBAPARAPI` environment variable (set in the previous step).

Other settings –

In addition, you must ensure that the default Java version is 1.7.0_55. This can be known by typing the `java -version` command. If for some reason the Java version is different, follow the below steps to set the correct Java version:

- Run `update-alternatives` and set the installed Oracle Java 7 to highest priority.
```
sudo update-alternatives --install "/usr/bin/java" "java"
"$JDKDIRECTORY/bin/java" 1
sudo update-alternatives --install "/usr/bin/javac" "javac"
"$JDKDIRECTORY/bin/javac" 1
sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"$JDKDIRECTORY/bin/javaws" 1
```
- Set the permissions.
```
sudo chmod a+x /usr/bin/java
```

```
sudo chmod a+x /usr/bin/javac
sudo chmod a+x /usr/bin/javaws
```

- Update to the highest priority link.
  `sudo update-alternatives --auto java`

Now check whether `java -version` returns the desired version.

3.5.2 How to run the pre-built samples

The prebuilt jar files for the Aparapi samples are located in the respective sample directory. For example, the Mandel prebuilt jar file for the Mandel sample is located under: `$<APPSDKSamplesInstallPath>/samples/aparapi/examples/Mandel/`.

To execute the sample, run the appropriate `<example>.sh` from the terminal. `./<example>.sh -h` lists the command-line options provided by the sample.

3.5.3 How to rebuild the samples

In order to build all the samples, run the `build.sh` batch file under `$<APPSDKSamplesInstallPath>/samples/aparapi`.

To build individual samples, go to the respective sample directory and type `ant`. For example, to compile the Mandel sample, run: `$<APPSDKSamplesInstallPath>/samples/aparapi/examples/Mandel> ant`

3.6 OpenCV

3.6.1 Prerequisites

In addition to the common prerequisites mentioned in Section 3.2, “General prerequisites”, to build and run the OpenCV-CL samples you must ensure the following installations and environment variable settings:

Installations –

- Download the OpenCV 2.4.9 source from: http://opencv.org/.
  The AMD APP SDK 2.9.1 OpenCV-CL samples work with OpenCV 2.4.9.
- Compile the downloaded source files to generate the OpenCV binaries, as described in Appendix D, “Building OpenCV from sources”. For answers to frequently asked questions on OpenCV, see the AMD APP SDK FAQ.
- Install the `Libgtk2.0-dev` and `pkg-config` packages.

Environment Variables –

Before building and running the AMD APP SDK OpenCV-CL samples, you must set the following environment variables after generating the OpenCV binaries. Assuming that the generated OpenCV binaries are generated under the `=/usr/local/` directory:

- Set and export `OPENCV_DIR` to the root directory containing the OpenCV include and lib files. For example, `OPENCV_DIR=/usr/local/`.

- Set and export `OCVCL_VER` to the OpenCV version used for the AMD APP SDK 2.9.1 release, i.e., set `OCVCL_VER = 249`.
- Set `LD_LIBRARY_PATH` to the directory containing the OpenCV library files. For example, assign `LD_LIBRARY_PATH` to `/usr/local/lib`.

3.6.2 How to run the pre-built samples

The AMD APP SDK ships with pre-built binaries of the OpenCV-CL samples. To execute the samples, you must perform the following steps:

- Open a terminal window.
- Change the directory to the appropriate architecture directory (x86 or x86_64) within *<<APPSDKSamplesInstallPath>>*`/samples/opencv/bin`.
- Run the samples by typing the name of the executables. You may need to prepend the name of the executable with `./`.
  To review the command line arguments for samples, see the respective sample documents.

3.6.3 How to rebuild the samples

To compile the AMD APP SDK OpenCV-CL samples, the CMake build tool is required. To build all the OpenCL samples, run the following commands:

```
$> cmake -G "Unix Makefiles"
"<APPSDKSamplesInstallPath>/samples/opencv/"
$> make
```

To build individual samples, run CMake from the respective sample source directory.

For more details on using CMake to generate make files for the AMD APP SDK samples, see Appendix C, “CMAKE”.

Appendix A Important Notes

- Unless specifically recommended otherwise, developers must use the latest graphics drivers for their platform. These drivers can be downloaded from http://support.amd.com/en-us/download.

  For current recommendations, see http://developer.amd.com/tools-and-sdks/opencl-zone/opencl-tools-sdks/amd-accelerated-parallel-processing-app-sdk/system-requirements-driver-compatibility/.

- The following values are returned when querying strings from OpenCL:

  `CL_PLATFORM_VERSION`: OpenCL 1.2 AMD-APP (*build.revision*)
  `CL_PLATFORM_NAME`: AMD Accelerated Parallel Processing
  `CL_PLATFORM_VENDOR`: Advanced Micro Devices, Inc.

Check the Platform Vendor string, not the Platform Name, to determine AMD hardware. For example code that shows how to check and use the `CL_PLATFORM_VENDOR` string, see the AMD APP OpenCL samples.

Appendix B Supported Devices

AMD is continually qualifying devices. For an up-to-date list of supported devices, please visit the APP SDK System Requirement and Driver Compatibility page.

Appendix C CMAKE

Getting started with CMake:

Starting with APP SDK 2.9, the build tool used in the APP SDK is CMake. CMake supports creating make files across different platforms and generating project files across different IDEs including Visual Studio and Eclipse. In order to use CMake with APP SDK, the CMake binaries must be downloaded.

1. CMake binaries can be found at: http://www.cmake.org/cmake/resources/software.html.
2. The following page describes how to build a project for Windows and Linux using CMake: http://www.cmake.org/cmake/help/runningcmake.html.

Quick links to download:

1. For Windows: Download the latest CMake from: http://www.cmake.org/cmake/resources/software.html.
2. For Linux: The "`sudo apt-get install cmake`" command will install the cmake binaries.

Note: The APP SDK 2.9.1 package for Windows includes Visual Studio 2010 and 2012 projects in addition to the CMake files. Developers can use the project files for building and for development. Visual Studio 2013 projects are not provided in the AMD APP SDK package. However, the CMake files of the samples can be used to generate VS 2013 project files.

Building an APP SDK sample using CMake:

Before using CMake to generate Visual Studio projects or makefiles for any sample, ensure that all the environment variables required for the particular sample are set. For example, if CMake is to be used for any Bolt sample, all the environment variables corresponding to Bolt headers and libraries must be set. The later sections in this document specify which environment variables must be set for each category of samples.

a. Windows: Generating Visual Studio project files –

Developers can create Visual Studio project files for the sample individually or for all of them together. To create the project files individually, CMake must be run by specifying the corresponding sample directory. For example, if you want to create the Visual studio 2012 solution file for the AtomicCounters OpenCL sample, then the corresponding command is:

For 32 bit:

```
cmake.exe -G "Visual Studio 11" "<path to the AtomicCounters sample source>"
```

For 64 bit:

```
cmake.exe -G "Visual Studio 11 Win64" "<path the to the AtomicCounters sample source>"
```

It is recommended to create another directory under the sample and run CMake from there. CMake creates a few additional files and projects apart from the main sample project. Hence running CMake from a sub-directory will keep the project files separate from the sample source files. If run from a subdirectory, the command will change as follows:

```
cmake.exe .. -G "Visual Studio 11" "<path to the sub-directory inside the AtomicCounters sample source folder>"
```

To build all the OpenCL samples, run the following command

For 32 bit:

```
cmake -G "Visual Studio 11" "<APPSDKSamplesInstallPath>/samples/opencl/"
```

For 64 bit:

```
cmake -G "Visual Studio 11 Win64" "<APPSDKSamplesInstallPath>/samples/opencl/"
```

Alternatively, `cmake-gui` can also be used to build the samples.

b. Linux: Generating Makefiles –

To generate the Makefiles, the same commands mentioned in the Windows section can be used except that the "Generator" name must be changed from the Visual Studio variant to the Unix variant.

For example, to build all the OpenCL samples, run the following command:

```
cmake -G "Unix Makefiles" "<APPSDKSamplesInstallPath>/samples/opencl/"
```

CMAKE APP SDK-SPECIFIC OPTIONS:

1. `BUILD_OPENCL`: ON/OFF - Builds OpenCL samples. Default is ON.

2. `BUILD_OPENCV`: ON/OFF - Builds OpenCV samples .Default is ON.
3. `BUILD_AMP`: ON/OFF - Builds C++ AMP samples. Default is ON.
4. `BUILD_BOLT`: ON/OFF - Builds Bolt samples. Default is ON.
5. `ENABLE_TBB`: ON/OFF - Builds all Bolt samples with the `multicore` option enabled. Default is OFF.

For example, to build all the samples except OpenCV samples, use `-DBUILD_OPENCV=OFF` as an argument to `cmake` or deselect the corresponding box in `cmake-gui`.

```
cmake -G "Unix Makefiles" -DBUILD_OPENCV=OFF <APPSDKSamplesInstallPath>
```

Note: The APP SDK samples have been tested with the following generators in CMake

1. Visual Studio 2010
2. Visual Studio 2012
3. Visual Studio 2013
4. NMake Makefiles
5. Unix Makefiles

Appendix D Building OpenCV from sources

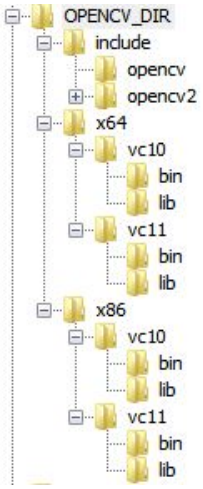
To build the OpenCV binaries from the sources, perform the following actions:

Building OpenCV from sources

To build the OpenCV binaries from the sources, perform the following actions:

1. Build the OpenCV library from the source files.
   The OpenCV sources for the 2.4.9 build are available at https://github.com/Itseez/opencv/tree/2.4.9. To have OpenCL support for OpenCV, the `opencv_ocl` library must be built. During configuring with CMake, select the `WITH_OPENCL` option and provide the path of the OpenCL library (`libOpenCL.so` in Linux and `OpenCL.lib` in Windows). The following links from opencv.org are useful:
   - Linux: http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html
   - Windows: http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html
2. Set the correct directory structure.
   On Windows, the directory structure of the complied OpenCV binaries created in the preceding step must be restructured as per the OpenCV prebuilt directory structure. Restructure the OpenCV binaries created from sources as per the prebuilt directory structure for Windows as shown in the following figure. Place the created `include` directly under the root OpenCV directory and copy the `bin` and `lib` files (both debug and release versions) into the following locations:
   - x86/vc10, if your binaries and libraries are built with Microsoft Visual Studio 10 and the target is x86.

- x86/vc11, if your binaries and libraries are built with Microsoft Visual Studio 12 and the target is x86.
- x86/vc12, if your binaries and libraries are built with Microsoft Visual Studio 13 and the target is x86.
- x64/vc10 if your binaries and libraries are built with Microsoft Visual Studio 10 and the target is x64.
- x64/vc11 if your binaries and libraries are built with Microsoft Visual Studio 12 and the target is x64.
- x64/vc12 if your binaries and libraries are built with Microsoft Visual Studio 13 and the target is x64.



Alternatively, you may skip restructuring the directory as per the OpenCV pre-builts, in which case, the Visual Studio property sheets of the OpenCV-CL projects must be updated to point to the correct paths.

For Linux, retain the default directory structure of the OpenCV binaries created from the sources.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:

URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Forum: developer.amd.com/openclforum