WHITE PAPER

High-Performance Computing Comprehensive Development Environment



A How-To Guide for Intel[®] SDK for OpenCL[™] Applications

Installing Intel[®] SDK for OpenCL[™] Applications for Linux* onto Ubuntu* 16.04 and getting the Intel[®] OpenCL code samples up and running in the Eclipse* IDE

Intel[®] SDK for OpenCL[™] Applications is a comprehensive environment for developing and optimizing OpenCL applications on Intel[®] platforms, and part of an increasingly rich portfolio of Intel tools for heterogeneous programming.

This paper explains how to install Intel for OpenCL Applications for Linux* onto Ubuntu* 16.04 and get the Intel® OpenCL code samples up and running in the Eclipse* integrated development environment (IDE).

We assume you already have Ubuntu set up with the Eclipse IDE for C++ development installed. Specifically, the environment includes:

- Ubuntu 16.04
- Eclipse Oxygen*

Installation

We did all the setup on a fresh, clean Ubuntu 16.04 build.

To set up the Ubuntu environment, we recommend you follow the instructions in Getting Started in Linux with Intel[®] SDK for OpenCL[™] Applications, which has two main parts:

- 1. Install the OpenCL code drivers. The first step is to install the OpenCL code drivers onto your Ubuntu system. This can be done using the script that is provided. Explaining the details of this script is beyond the scope of this document.
- 2. Install Intel SDK for OpenCL Applications. On page 2, you will find a script that installs the Intel SDK for OpenCL applications. This script performs all the necessary steps to put the SDK in the proper locations.

These two scripts can take a little while to run. Once the run completes, the Intel SDK for OpenCL applications has been installed in the location shown in **Figure 1**.

Getting the OpenCL Code Samples

Intel has several OpenCL code samples for both Windows* and Linux at Intel® SDK for OpenCL[™] Applications Support. If you scroll, you will see a link to download the Linux samples, or you can click "OpenCL 1.2 samples."

What Gets Extracted

After extracting your samples onto the hard drive, you should see something similar to **Figure 2**. We placed our samples in Home/Dev/IntelOpenCLSamples.

Contents

Installation1
Getting the OpenCL Code Samples 1
Example Descriptions2
The Common Folder 2
Creating the Projects in Eclipse:
Oxygen 3
Create a New Eclipse Project 3
Project File References7

White Paper | Intel[®] SDK for OpenCL[™] Applications

😣 🖨 🗉 opencl				
< > 🖻 opt i	ntel opencl include		ୟ ∷≣	
⊘ Recent	Name	 Size 	Туре	Modif
✿ Home	include			
🖿 Desktop	clbltfne9.rtl	1.5 MB	Program	Nov 1
Documents	clbltfne9_img_cbk.o	373.3 kB	Document	Nov 1
🕹 Downloads	clbltfne9_img_cbk.rtl	603.3 kB	Program	Nov 1
Husic	clbltfnh8.rtl	1.6 MB	Program	Nov 1
Pictures	clbltfnh8_img_cbk.o	389.9 kB	Document	Nov 1
Videos	clbltfnh8_img_cbk.rtl	606.1 kB	Program	Nov 1
D Trash	clbltfnl9.rtl	1.1 MB	Program	Nov 1
⊡ Network	clbltfnl9_img_cbk.o	325.8 kB	Document	Nov 1
Computer	clbltfnl9_img_cbk.rtl	549.5 kB	Program	Nov 1
Connect to Server	clbltfnshared.rtl	1.1 MB	Program	Nov 1
	igdclbif.bin	8.1 MB	Binary	Nov 1
	kernel-4.4-xcode.patch	877.5 kB	Text	Nov 1
	kernel-4.7.patch	126.3 kB	Text	Nov 1
	libclang_compiler.so	4.9 MB	Unknown	Nov 1
	libcl_logger.so	149.7 kB	Unknown	Nov 1
	libcommon_clang.so	47.0 MB	Unknown	Nov 1
	libcommon_clang_legacy.so	39.7 MB	Unknown	Nov 1
	libcpu_device.so	427.0 kB nclude" select	Unknown ed (containing 2 ite	Nov 1 ems)

Figure 1. Intel SDK for OpenCL Applications installation

😣 🗇 🕕 IntelOpenCLSamj	ples			
く 〉 企Home Dev	IntelOpenCLSamples			۹ 🗉 🗉
O Recent	Name	 Size 	Туре	Modified
✿ Home	BitonicSort	17 items	Folder	10:00
E Desktop	CapsBasic	13 items	Folder	10:00
Documents	common	11 items	Folder	Nov 19
Downloads	GEMM	16 items	Folder	10:00
J Music	GodRays	19 items	Folder	10:00
Pictures	MedianFilter	17 items	Folder	10:00
Videos	MonteCarlo	16 items	Folder	10:00
Trash	MotionEstimation	11 items	Folder	10:00
Dr Network	MotionEstimationAdvanced	11 items	Folder	10:00
Computer	MultiDeviceBasic	22 items	Folder	10:00
Connect to Server	ProcGraphicsOpt	15 items	Folder	Nov 13
	SimpleOptimizations	17 items	Folder	10:00
	templates 1	4 items	Folder	Nov 13
	ToneMapping	19 items	Folder	10:01
	ToneMappingMultiDevice	18 items	Folder	10:01
	Makefile	3.3 kB	Text	Nov 26 2015
	OpenCLSamples_2012.sln	9.1 kB	Text	Nov 26 2015
	OpenCLSamples_2013.sln	9.0 kB	Text	Nov 26 2015
	OpenCLSamples_2015.sln	9.2 kB	Text	Nov 26 2015
	README.TXT	7.2 kB	Text	Nov 26 2015
	README_LIN.TXT	6.5 kB	Text	Nov 26 2015
	README_WIN.TXT	7.2 kB	Text	Nov 26 2015

Figure 2. Extracting samples onto the hard drive

Example Descriptions

Each sample folder has a .PDF file that gives you a full description of each project. Here's a brief description of each sample:

 BitonicSort. Demonstrates how to sort an arbitrary input array of integer values with OpenCL software technology using Single Instruction Multiple Data (SIMD) bitonic sorting networks.

- **CapsBasic.** Demonstrates how to query all OpenCL platforms available on the system and lists all devices for a given platform.
- GEMM (General Matrix Multiply). Demonstrates how to efficiently utilize an OpenCL device to perform a general matrix multiply operation on two dense square matrices.
- **GodRays.** Demonstrates how to use high dynamic range (HDR) rendering with the God Rays (crepuscular rays) effect in OpenCL software technology.
- MedianFilter. Demonstrates how to use medial filter in OpenCL software technology by optimizing the filtration process using implicit single instruction, multiple data (SIMD) code vectorization performed by the built-in OpenCL software technology compiler vectorizer.
- MonteCarlo. Demonstrates implementation of the Monte Carlo simulation for the European stock option pricing.
- MotionEstimation. Provides step-by-step guidelines on using Intel's motion estimation extension for OpenCL software technology.
- **MotionEstimation Advanced.** Expands motion estimation by including a set of host-callable functions for frame-based video motion estimation.
- **MultiDeviceBasic.** Sample utilizes the capabilities of the multidevice system, CPU/GPU.
- ProGrapicsOpt. Demonstrates how to optimize OpenCL software technology kernels for running on graphics devices with the Intel[®] Processor Graphics optimization sample based on the Sobel Filter* algorithm.
- SimpleOptimizations. Demonstrates simple ways to measure the performance of OpenCL software technology kernels in an application.
- ToneMapping. Demonstrates how to use HDR rendering with the tone mapping effect in OpenCL software technology.

The samples come ready to be compiled on either Windows or Linux. At the root of the entire samples folder, and in each individual project folder, are Visual Studio* files that are not needed for the Linux environment. We removed them, since they were unnecessary.

The Common Folder

The "common" folder contains wrapper .hpp and .cpp source code files that aid in the sample apps. These are specific to the SDK samples and are not part of the Intel SDK for OpenCL Applications library itself.

While it may not be necessary, we copied the common folder to the default Intel SDK for OpenCL Applications install location and put it in the Includes folder (**Figure 3**). This is because we wanted to better organize our projects and have just one main include directory. In Eclipse, we reference this location in every project. The rationale is that this is Intel's implementation of OpenCL software technology and samples, and some of the extra classes created in the common folder could be useful in other projects.

White Paper | Intel[®] SDK for OpenCL[™] Applications

😣 🖨 🗊 include			
< > 🖻 opt in	ntel opencl include CL		Q := :::
⊘ Recent	Name	▲ Size	Type Modif
✿ Home	CL CL	10 items	Folder Nov 1
🛅 Desktop	common	11 items	Folder 15:43
Documents			
Downloads			
J Music			
Pictures			
Videos			
🗑 Trash			
Detwork			
Computer			
Connect to Server			
Figure 3. Copyi	ng the common f	older to the d	efault Intel SDK

C++ Project Create C++ project of selected type Project name: SimpleOptimization Use default location Location: /home/rick/Dev/eclipse-workspace/SimpleOptimization Browse... Choose file system: default 🔻 Toolchains: Project type: GNU Autotools Cross GCC 🕶 🗁 Executable Empty Project Hello World C++ Project Shared Library Static Library Makefile project Show project types and toolchains only if they are supported on the platform ? < Back Next > Cancel Finish

Figure 3. Copying the common folder to the default Intel SDK for OpenCL applications install location and putting it in the Includes folder

Creating the Projects in Eclipse: Oxygen*

Assumptions

As mentioned, Oxygen is the current version that's available at the time of this writing. We also assume you have a basic knowledge of using the IDE and know how to create a C++ project and set the Eclipse workspace.

Overview of the Process

There are 14 samples, each of which will be converted into an Eclipse project. They all follow the same basic pattern:

- **Create a new Eclipse C++ project** with the same name as the original source sample.
- Set up the properties for the project: Header file location, OpenCL lib path, and so on.
- Create a "Src" folder and copy in all required .CPP files.
- Create a "Resources" folder for any extra files needed per sample project.
- **Rebuild** the Project C/C++ Index.
- Compile and fix any issues.

Create a New Eclipse Project

Step 1: Create a New Project

From the File menu, choose "New C/C++ project." In **Figure 4**, you can see we were working with the "SimpleOptimization" project. We chose an empty project with Linux GCC.

Click "next." The Select Configurations dialog box (Figure 5) appears.

Figure 4. Choose an Empty Project with Linux GCC



Figure 5. Select Configurations dialog box

Click "Advanced Settings." The Properties dialog box for the Eclipse project appears.

It's not mandatory to click "Advanced Settings" at this stage to fill out the project properties. You can instead click "Finish," and, once the project is created, enter this information later by right-clicking a project and choosing "Properties."

Step 2: Add the Include File Paths

Add the paths to both the OpenCL software technology header files (**Figure 6**). Recall that we copied the entire common folder from the samples folder to be in the same location as the Intel SDK for OpenCL applications installation location. You may or may not have done this already, but you will need to add a reference to the common folder no matter where you have it on your hard drive.

Step 3: Add the OpenCL Software Technology Library and Path

You need to add both the OpenCL software technology library itself and the path where Eclipse should look for the library (**Figure 7**).

Step 4: Choose Compiler Options

We added the -std=c++11 flag under "Miscellaneous" settings (**Figure 8**). We found that some code sample applications needed this, so we added it to all of the projects.



Figure 6. Adding the paths to both the OpenCL software technology header files



Figure 7. Adding the OpenCL software technology library and the library search path

-yp	Securitys	$(\phi + \phi + \phi)$
Resource Builders C/C++ Build Build Variables	Configuration: Debug [Acti	we] Manage Configurations Manage Configurations Second Secon
Linvionnent Logging Settings Tool Chain Editor C/C++ General Linux Tools Path Project Réferences Run/Doebug Settings Task Rags Validation WikiText	B) GCC C++ Compiler Dialect Preprocessor Micludes Optimization Debugging Warnings Miscellaneous Optimization Dialect Dialect Dialect Dialect Dialect Distribution Debugging Warnings Miscellaneous Miscellaneous Wiscellaneous Shared Libraries Miscellaneous Shared Libraries Shared Library Settings Scaembler @ General	Other flags _c-fmessage-length=0-std=c++11 Verbose (-v) Position independent Code (-fPIC) Support for pthread (-pthread)



Step 5: Post Build Command

This step is not required. It copies everything from the Resources folder into the Debug folder so that once the compile is done, you can jump into a terminal, navigate to the executable, and have everything ready to go in the proper folder (**Figure 9**).

Once you have completed filling in the project properties, click "Apply and close." The Select Configurations dialog box appears. Click "Finish." You now have an empty project set up and ready to go.

Step 6: Create a Src Folder

At this point, there is an empty project with no source code files. By convention, it's common to put source code files in a Src folder. Next, you need to create a new folder called Src that will hold all the .CPP files.

- Right-click the name of the project.
- Click "New" and then "Folder.
- Name the new folder Src.

Your project now has a Src folder under the root.

 Pesource Builders Configuration: Debug [Active] Manage Configuration: Debug [Active] Probuild Steps Command: Description: Description: Description: Description: Description: Description: 	pe filter text 🛛 🛛	Settings	⇔ • ⇔ •
Build Variables Environment Environment Environment Logging Pre-build steps Settings Pre-build steps Command: Command: Validation Vertices* WikText Description: Description: Command: Command: Command: Command: Command:	esource uilders /C++ Build	Configuration: Debug [Active]	Manage Configurations.
Linux Tools Path Project References Run/Debug Settings Task Repository Validation WikiText Description: Command: Cp/resources/*. Description:	Build Variables Environment Logging Settings Tool Chain Editor /C++ General	Tool Settings	C Error Parsers
Task Tags Validation WikiText Command: Cp/resources/*. Description:	inux Tools Path roject References un/Debug Settings ask Repository	Post-build steps	¥
	ask Tags alidation /ikiText	Command: cp/resources/*.	•
		Description.	•
Deskere Defaulte			va Dafaultz) 🔽 A = h:
Restore Defaults A		Rest	ore <u>D</u> efaults <u>Apply</u>

Figure 9. Optional Post Build command

▼ [™] SimpleOptimizations	
▶ 🗊 Includes	
🕨 🗁 Debug	
🕨 🗁 resources	
* 备 SFC	

Figure 10. Creating a src folder

Step 7: Copy the Application and Supporting CPP Files

Now that you have a folder to hold the CPP files, we need to copy the existing sample files into this new Src folder. This is as simple as navigating to the original Samples folder for a given project, grabbing the .CPP files, and dropping them into the Eclipse IDE's corresponding project (**Figure 11**).

8	🖲 🗊 SimpleOptimi:	zations						
	> < IntelOpe	enCLSamples	SimpleOptimizations			٩		
0	Recent	Name		^	Size	Туре	Modifie	d
企	Home	Makel	file		538 bytes	Text	Nov 26 2	2015
	Desktop	Simple	eOptimizations		296.3 kB	Program	Nov 27	
۵	Documents	C Simple	eOptimizations.cl		1.6 kB	Text	Nov 26 2	2015
∻	Downloads	Simple	eOptimizations.cpp		13.6 kB	Text	Nov 26 2	2015
9	Music	stdaf	cpp		300 bytes	Text	Nov 26 2	2015
Ø	Pictures	stdaf:	c.h		383 bytes	Text	Nov 26 2	2015
н	Videos	targel	tver.h		763 bytes	Text	Nov 26 2	2015
1	Trash	user_e	guide.pdf		97.6 kB	Document	Nov 26 2	2015
ō,	Network							
9	Computer							
¢,	Seagate Bac 🔺							
모	Connect to Server							

Figure 11. Copying the application and supporting .CPP files

In Figure 11, there is only one .CPP file that we need, the SimpleOptimizations.cpp file, which happens to be the main app source code file.

Step 8: Copy the CPP Files from the Common Folder

Next, you need to determine which .CPP files from the Samples common folder you need to copy into the Src folder. To do this, look at the main source code file (in this case, the SimpleOptimizations.cpp file show in **Figure 12**).

White Paper | Intel[®] SDK for OpenCL[™] Applications

21 22 23	#include #include	<iostream> <stdio.h></stdio.h></iostream>
24	#ifndef	linux
25	<pre>#include</pre>	"stdafx.h"
26	#else	
27	<pre>#include</pre>	"math.h"
28	#endif	
29		
30	<pre>#include</pre>	"basic.hpp"
31	<pre>#include</pre>	"cmdparser.hpp"
32	<pre>#include</pre>	"oclobject.hpp"
33		- · · ·

Figure 12. Determining which .CPP files from the Samples common folder to copy into the Src folder

Looking at Figure 12, you can see that this file relies on Basic.hpp, Cmdparser.hpp, and Oclobject.hpp. This means you need to import the corresponding .CPP files from the Common folder and place them into the project's Src folder.

As previously mentioned, we put a copy of the Common folder alongside the Intel SDK for OpenCL Applications include folder, which is located at: /opt/intel/opencl/include.

After grabbing all the files needed for the project to compile properly, you will end up with a Src folder that looks like **Figure 13**.

だ SimpleOptimizations

- Binaries
- Debug
- resources
- ▼_Bsrc
 - Ising the second sec
 - Condparser.cpp
 - Ic oclobject.cpp
 - SimpleOptimizations.cpp
 - Lo utils.cpp

Figure 13. Src folder

Step 9: Create the "Resources" Folder

This folder contains the .CL kernel files, images, and any other files that are needed at the application's runtime. Collecting all the files into one folder makes it easy for the post-build process to easily find them to copy into the Debug folder.

Not all projects have extra files that need to run. If not, you can omit this step for that project. Otherwise, just as we did with the Src folder, right-click the project, click "New" and then "Folder" and then name it "Resources."

Now we have to go back to the original Samples project folder. For this example, we can see that there is only one additional file, SimpleOptimizations.CL (Figure 14).

😕 🖨 🗊 SimpleOptimi	zations				
< > < IntelOp	enCLSamples SimpleOptimizations			c	λ ∷≣ ∷∷
⊘ Recent	Name	•	Size	Туре	Modified
✿ Home	Makefile		538 bytes	Text	Nov 26 2015
🛅 Desktop	SimpleOptimizations		296.3 kB	Program	Nov 27
Documents	SimpleOptimizations.cl		1.6 kB	Text	Nov 26 2015
🕹 Downloads	SimpleOptimizations.cpp		13.6 kB	Text	Nov 26 2015
J Music	stdafx.cpp		300 bytes	Text	Nov 26 2015
Pictures	stdafx.h		383 bytes	Text	Nov 26 2015
Videos	targetver.h		763 bytes	Text	Nov 26 2015
🗑 Trash	user_guide.pdf		97.6 kB	Document	Nov 26 2015
Network					
Computer					
🔇 Seagate Bac 🔺					
Connect to Server					

Figure 14. SimpleOptimizations.CL folder

Copy SimpleOptimizations.CL into the Eclipse project's Resource folder. You should now have a Resources folder that looks like **Figure 15**.

Step 10: Re-Index C++

Because the Eclipse IDE has syntax completion similar to Visual Studio, sometimes it needs to be re-indexed so that it knows about new source code. To do this, from the Project menu, click "C/C++ Index" and then "Rebuild" (Figure 16).

▼SimpleOptimizations SimpleSimp
▶ 🗊 Includes
🕨 🗁 Debug
resources
SimpleOptimizations.cl
▶ 😹 src

Figure 15. Resources folder



Figure 16. Re-indexing C++

Project File References

Here are some useful links that discuss how to set up Eclipse properly that so it can detect and know about the new source code headers and classes:

- https://help.eclipse.org/oxygen/index.jsp?topic=%2Forg. eclipse.cdt.doc.user%2Freference%2Fcdt_u_prop_ general_sd_entries.htm
- https://www.eclipse.org/community/eclipse_ newsletter/2013/october/article4.php

Below is a list of the files we added to each project Src and Resources folder. This list includes only the supporting files needed for each project and not the actual project main source code files.

At the time of this writing, every sample application compiled and ran with the exception of MotionEstimation* and MotionEstimationAdvanced*. We suspect this is due to the fact that we have an Nvidia* graphics card in our system and have not downloaded the proper Nvidia drivers. On a system with Intel graphic technology, this should run.

NOTE: We're not sure why the original developers put the cmdoptions.hpp/cpp file in the original project rather than in the common folder. However, because there are a couple of projects that use these files, we left things as they were and left them in each individual folder's Src location.

BitonicSort

- Src: BitonicSort.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp
- Resources: BitonicSort.cl

CapsBasic

- Src: CapsBasic.cpp
- Resources: N/A

GEMM

- **Src:** Gemm.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp, cmdoptions.cpp, cmdoptions.hpp
- Resources: gemm.cl

GodRays

- Src: GodRays.cpp, basic.cpp, cmdparser.cpp, GodRaysNative.cpp, oclobject.cpp, stdafx.cpp, utils.cpp
- **Resources:** GodRays.cl, GodRays.rgb

MedianFilter

• **Src:** MedianFilter.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp

• Resources: MedianFilter.cl

MonteCarlo

- **Src:** MonteCarlo.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp, cmdoptions.cpp
- Resources: MonteCarlo.cl

MotionEstimation

- Src: basic.cpp, cmdparser.cpp, MotionEstimationAdvanced.cpp, oclobject.cpp, utils.cpp, yuv_utils.cpp
- Resources: video_1920x1080_5frames.yuv

MotionEstimationAdvanced

- Src: basic.cpp, cmdparser.cpp, MotionEstimation.cpp, oclobject.cpp, utils.cpp, yuv_utils.cpp
- Resources: mea_video_1920x1080_5Frames.yuv

MultiDeviceBasic

- **Src:** basic.cpp, cmdparser.cpp, kernel.cpp, multi.cpp, multidevice.cpp, oclobject.cpp, shared.cpp, system.cpp, utils.cpp
- Resources: cpu+mic.system-level.sh, cpu+multimic. system-level.sh, multimic.system-level.sh, universal. system-level.sh

ProGraphicsOpt

- **Src:** ProGraphicsOpt.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, stdafx.cpp, utils.cpp
- Resources: ProGraphicsOpt.cl

SimpleOptimizations

- **Src:** SimpleOptimizations.cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp
- Resources: impleOptimizations.cl

ToneMapping

- **Src:** ToneMapping.cpp, ToneMappingNative.cpp, basic. cpp, cmdparser.cpp, oclobject.cpp, utils.cpp
- Resources: ToneMapping.cl, ToneMapping.rgb

ToneMappingMultiDevice

- **Src:** oneMappingMultiDevice.cpp, ToneMappingNative. cpp, basic.cpp, cmdparser.cpp, oclobject.cpp, utils.cpp
- **Resource:** oneMappingMultiDevice.cl, ToneMappingMultiDevice.rgb



Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance etests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting www.intel.com/design/literature.htm. This sample source code is released under the Intel Sample Source Code License Agreement.

For more information regarding performance and optimization choices in Intel® Software Development Products, see our Optimization Notice: https://software.intel.com/articles/optimizationnotice#opt

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

Copyright © 2018 Intel Corporation Printed in USA 0218/SS