

1 Overview

1.1 Location `$<APPSDKSamplesInstallPath>\samples\opencl\cl\`

1.2 How to Run See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and at `$<APPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds.

Ensure that the OpenCL 2.0 environment is installed.

Type the following command(s).

1. `SimplePipe`
This runs the program with the default options.
2. `SimplePipe -h`
This prints the help file.

1.3 Command Line Options Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
-h	--help	Shows all command options and their respective meanings.
	--device	Devices on which the program is to be run. Acceptable values are cpu or gpu.
-q	--quiet	Quiet mode. Suppresses most text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing related statistics.
	--dump	Dump binary image for all devices.
	--load	Load binary image and execute on device.
	--flags	Specify compiler flags to build the kernel.
-p	platformId	Select platformId to be used (0 to N-1, where N is the number of available platforms).
-d	deviceId	Select deviceId to be used (0 to N-1, where N is the number of available devices).
-v	version	AMD APP SDK version string.
-i	iterations	Number of iterations for kernel execution.

Short Form	Long Form	Description
-x	--numPackets	Total Number of Packets to communicate between two kernels using Pipe
		-w --workgroups Number of work-group per kernel execution -l --localsize Number of work-items per work-group (should be 2^N)
-y	--packetSize	PacketSize in Bytes
-mp	--multiPipe	A flag indicating singlePipe or multiPipe use cases: 0 - SinglePipe [default] 1 - MultiPipe
-type	--kernelType	Type of built-in Pipe functions: 0 - reserve_read/write_pipe [default] 1 - work_group_reserve_read/write_pipe 2 - Convenience [without using reserve built-in pipe functions]
-w	--workgroups	Number of work-groups per kernel execution.
-l	--localsize	Number of work-items per work-group (should be 2^N)

2 Introduction

This sample demonstrates how to use Pipe memory objects. Pipe is a new feature introduced in the OpenCL 2.0 specification. Conceptually, the Pipe memory object is an ordered sequence of data items. A pipe has two endpoints: a write endpoint into which data items are inserted, and a read endpoint from which data items are removed.

The sample uses `clCreatePipe` to create Pipe objects. It runs on OpenCL 2.0 compliant devices.

3 Implementation Details

This sample demonstrates how to use various built-in Pipe functions, which is introduced in OpenCL 2.0 C Programming Language.

The sample illustrates two primary operations:

- Single Pipe Read-Write Uses: One kernel writes the input data into a Pipe and another kernel reads data from same Pipe.
- Multiple Pipe Read-Write Uses: One kernel writes the input data into multiple pipes and another kernel reads the data from multiple Pipes. (**Note:** To demonstrate the multi-pipe use case, this sample uses 4 pipes.)

OpenCL 2.0 has various built-in pipe functions. These functions are categorized based on when the actual read from or write to a pipe is performed, which is ensured using the `commit()` call.

This sample implements three basic kernels for each of the above primary operations:

a. **Work-item based Pipe Read and Write Built-in Function:** Using these built-in functions, read from or write to a pipe is performed at work-item level. At the work-item level, read from or write to a pipe can be performed in following two ways:

- **With reservation:** The sample uses this kernel as a default. This kernel first reserves a few packet entries for reading from or writing to a pipe and then calls the `commit_read/write_pipe` function to perform read from or write to pipe.
- **Without reservation:** This kernel does not reserve packet entries to perform read from or write to the pipe.

b. **Work-group based Pipe Read and Write Built-in Function:** Using these built-in functions, actual read from or write to pipe is performed at the work-group level. Since the actual read from or write to pipe is performed at the work-group level, this kernel gives better performance compared to work-item based reservation. The work-group level reservation also ensures that the entries reserved for writing are all added *in-order* as one contiguous set of packets to the pipe. These built-ins can be used to read from or write to a packet index either once or multiple times; if multiple work-items in a work group need to access a pipe item multiple times, this procedure is clearly beneficial.

4 References

1. [The OpenCL Specification, version 2.0, rev 22 document](#) (page 34).
2. [The OpenCL C Programming Language \(ver 2.0, rev 22\) document](#).

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



The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2014 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.