AMD **Accelerated** Parallel Processing TECHNOLOGY

# SAMPLE

# Pipe Producer Consumer Kernels

## 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. `PipeProducerConsumerKernels`
   This runs the program with the default options.
2. `PipeProducerConsumerKernels -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. |

| Short Form | Long Form | Description |
|---|---|---|
| -i | iterations | Number of iterations for kernel execution. |
| -s | --pipeSize | Pipe size in packets. |
| -b | --seed | Seed for random number generation. |

## 2 Introduction

This sample demonstrates the “pipe” memory object introduced in OpenCL 2.0, as a data sharing FIFO for a producer and a consumer kernel. A pipe object is a FIFO in which one kernel can write the data to and another kernel can read the data from in FIFO fashion.

This sample must be run in the OpenCL 2.0 environment.

## 3 Implementation Details

This sample implements two kernels -- one producer and another consumer, sharing data by means of a pipe. The producer kernel is a Parks-Miller random number generator. It also uses the Box-Muller transform to convert uniform random numbers to Gaussian distributed random numbers. The producer kernel generates and writes random numbers to a pipe. At the other end of the pipe, a consumer kernel reads these random numbers and generates their histogram.

## 4 References


1. The OpenCL Specification, version 2.0, rev 22 document.
2. The OpenCL C Programming Language (ver 2.0, rev 22) document.
3. Numerical Recipes in FORTRAN 77: The Art of Scientific Computing (Vol 1), Cambridge University Press, 1997.



**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