OpenCL C++ Wrapper 1.2 Reference Card - Page 1

OpenCL (Open Computing Language) is a multi-vendor open standard for general-purpose parallel programming of heterogeneous systems that include CPUs, GPUs, and other processors. OpenCL provides a uniform programming environment for software developers to write efficient, portable code for high-performance compute servers, desktop computer systems, and handheld devices.

Specifications and more information about OpenCL and the OpenCL C++ Wrapper are available at www.khronos.org.

The OpenCL C++ Wrapper is designed to be built on top of the OpenCL 1.2 C API and is not a replacement. The C++ Wrapper API corresponds closely to the underlying C API and introduces no additional execution overhead.

- [n.n.n] refers to a section in the OpenCL C++ Wrapper API Specification
- [n.n.n] refers to a section in the OpenCL API Specification
- type refers to the types for indicated parameters
- FunctionName refers to non-member functions

**cl::Platform**

- **Class**: cl::Platform is an interface to OpenCL cl_platform_id.
- **Function**:
  - `cl_int getPlatformInfo(const cl_platform_id & platform_id, void * user_data = NULL, cl_context_properties * properties = NULL);
  - `void (CL_CALLBACK * pfn_notify)(const char * user_data = NULL);
  - `cl_platform_id getPlatformInfo(void);
  - `cl_platform_id getPlatformId(void);
  - `cl_platform_id unloadPlatformCompiler(void);

**cl::Context**

- **Class**: cl::Context is an interface to OpenCL cl_context.
- **Function**: createContext
  - `cl_int createContext(const Context & context);
  - `cl_int createContextFlags(const Context & context, cl_context_properties flags);
  - `cl_int createContextFromType(void);

**cl::Device**

- **Class**: cl::Device is an interface to OpenCL cl_device_id.
- **Function**:
  - `cl_int getDeviceInfo(const Device & device, void * user_data = NULL, cl_int err = NULL);`
template<typename T> cl_int getInfo(const Context &context, cl_int *err = NULL);

Calls OpenCL function clGetCommandQueueInfo() [5.7.2]

---

declare(int x);

class cl::CommandQueue:

- constructor:
  - cl::CommandQueue(const Device &device, cl_command_queue_properties properties = 0, cl_int *err = NULL);
  - Calls OpenCL function clCreateCommandQueue() [5.1]

- getCommandGroupInfo:
  - template<typename T> cl_int getInfo(const CommandQueue &commandQueue, cl_int *err = NULL);
  - Calls OpenCL function clGetCommandQueueInfo() [5.7.2]

---

declare(int x);

class cl::UserEvent:

- constructor:
  - cl::UserEvent();
  - cl::UserEvent(Context &context, cl_int *err = NULL);

- static methods:
  - cl_int setEventCallback(cl_int type, void (CL_CALLBACK *fn)(cl_event event), cl_int *err = NULL);
  - void wait(cl_int *err = NULL);

- member methods:
  - template<typename T> cl_int getProfilingInfo(cl_int *err = NULL);
  - Calls OpenCL function clGetEventProfilingInfo() [5.12]

---

declare(int x);

class cl::UserEvent:

- constructor:
  - cl::UserEvent();
  - cl::UserEvent(Context &context);

- static methods:
  - cl_int setEventCallback(cl_int type, void (CL_CALLBACK *fn)(cl_event event), cl_int *err = NULL);
  - void wait(cl_int *err = NULL);

- member methods:
  - template<typename T> cl_int getProfilingInfo(cl_int *err = NULL);
  - Calls OpenCL function clGetEventProfilingInfo() [5.12]

---

declare(int x);

class cl::Kernel:

- constructor:
  - cl::Kernel(const Device &device, const size_t *param);
  - Calls OpenCL function clCreateKernel() [5.2.2]

- method:
  - void enqueueWriteBufferRect(cl_int *err = NULL);
  - Calls OpenCL function clEnqueueWriteBufferRect() [5.2.2]

---

declare(int x);

class cl::UserEvent:

- constructor:
  - cl::UserEvent();
  - cl::UserEvent(Context &context);

- static methods:
  - cl_int setEventCallback(cl_int type, void (CL_CALLBACK *fn)(cl_event event), cl_int *err = NULL);
  - void wait(cl_int *err = NULL);

- member methods:
  - template<typename T> cl_int getProfilingInfo(cl_int *err = NULL);
  - Calls OpenCL function clGetEventProfilingInfo() [5.12]

---

declare(int x);

class cl::Kernel:

- constructor:
  - cl::Kernel(const Device &device, cl_kernel_work_group_info *info, cl_int *err = NULL);
  - Calls OpenCL function clCreateKernel() [5.12]

- method:
  - cl::Kernel enqueueWriteBufferRect(cl_int *err = NULL);
  - Calls OpenCL function clEnqueueWriteBufferRect() [5.2.2]

---

declare(int x);

class cl::Context:

- constructor:
  - cl::Context()

- member methods:
  - cl_int::size_t getDeviceIDs(const Device &device, cl_device_type type, void *ptr, cl_int *err = NULL);
  - Calls OpenCL function clGetDeviceIDs() [5.7.2]

---

declare(int x);

class cl::CommandQueue:

- constructor:
  - cl::CommandQueue(const Device &device, cl_command_queue_properties properties = 0, cl_int *err = NULL);
  - Calls OpenCL function clCreateCommandQueue() [5.1]

- method:
  - cl::CommandQueue enqueueWriteBufferRect(cl_int *err = NULL);
  - Calls OpenCL function clEnqueueWriteBufferRect() [5.2.2]

---

declare(int x);

class cl::Buffer:

- constructor:
  - cl::Buffer(const Device &device, size_t *ptr, cl_int *err = NULL);
  - Calls OpenCL function clCreateBuffer() [5.2.2]

- method:
  - cl::Buffer enqueueWriteBufferRect(cl_int *err = NULL);
  - Calls OpenCL function clEnqueueWriteBufferRect() [5.2.2]
```cpp
// 5.2.2

// Calls OpenCL function clEnqueueCopyBufferRect.

// Calls OpenCL function clFinish.

// Calls OpenCL function clEnqueueMapImage.

// Calls OpenCL function clEnqueueUnmapMemObject.

// Calls OpenCL function clEnqueueTask.

// Calls OpenCL function clEnqueueNativeKernel.

// Calls OpenCL function clEnqueueMigrateMemObjects.

// Calls OpenCL function clEnqueueMarkerWithWaitList.

// Calls OpenCL function clEnqueueBarrierWithWaitList.

// Calls OpenCL function clFlush.

// Calls OpenCL function clFinish.

// Calls OpenCL function clEnqueueAcquireGLObjects.

// Calls OpenCL function clEnqueueReleaseGLObjects.

// The following functions are also available as non-member functions, where they operate on the default command queue:

enqueueReadBuffer()    enqueueWriteBuffer()
enqueueCopyBuffer()    enqueueCopyImageToBuffer()
enqueueFillImage()     enqueueFillImageToBuffer()
enqueueNativeKernel()  enqueueMigrateMemObjects()
enqueueTask()          enqueueWaitList()
OpenCL C++ Wrapper 1.2 Reference Card

Functors

cl::EnqueueArgs
structure cl::EnqueueArgs is an interface for dispatching
EnqueueArgs::NDRange global;
EnqueueArgs::NDRange global, NDRange local;
EnqueueArgs::Queue commandQueue, NDRange global;
EnqueueArgs::CommandQueue commandQueue, NDRange global, NDRange local;

cl::make_kernel
structure template<class... Args> cl::make_kernel is an interface for exporting a functor interface to

cl::KernelFunctorGlobal
class template<class... Args> KernelFunctorGlobal is an interface for exporting a functor interface to

Exceptions [4]
To enable the use of exceptions, define the preprocessor macro __CL_ENABLE_EXCEPTIONS. Once enabled, an error
originally reported via a return value will be reported by
throwing an exception of type std::exception.

Preprocessor macro names:

Supported Data Types
The optional double scalar and vector types are supported if

Built-in OpenCL Scalar Data Types [6.1.1]

OpenCL Type | API Type | Description
--- | --- | ---
bool | -- | true (1) or false (0)
char | cl_char | 8-bit signed
unsigned char, uchar | cl_uchar | 8-bit unsigned
short | cl_short | 16-bit signed
unsigned short, ushort | cl_ushort | 16-bit unsigned
int | cl_int | 32-bit signed
unsigned int, uint | cl_uint | 32-bit unsigned
long | cl_long | 64-bit signed
unsigned long, ulong | cl_ulong | 64-bit unsigned
float | cl_float | 32-bit float
double | cl_double | 64-bit, IEEE 754
half | cl_half | 16-bit float (storage only)
sizet | -- | 32- or 64-bit unsigned integer

Built-in OpenCL Vector Data Types [6.1.2]

OpenCL Type | API Type | Description
--- | --- | ---
charn | cl_charn | 8-bit unsigned
uchar:n | cl_uchar:n | n-byte unsigned
short:n | cl_short:n | n-byte signed
ushort:n | cl_ushort:n | n-byte unsigned
int:n | cl_int:n | n-byte signed
uint:n | cl_uint:n | n-byte unsigned
long:n | cl_long:n | n-byte signed
ulong:n | cl_ulong:n | n-byte unsigned
floatn | cl_float:n | n-byte float
doublen | cl_doublen | n-byte double
halfn | cl_halfn | n-byte half

OpenCL Reserved Data Types [6.1.4]

OpenCL Type | Description
--- | ---
booln | boolean vector
halfn | 16-bit, vector
quad, quadn | 128-bit float, vector
complex half, complex halfn imaginary half, imaginary halfn | 16-bit complex, vector
complex float, complex floatn imaginary float, imaginary floatn | 32-bit complex, vector
complex double, complex doublen imaginary double, imaginary doublen | 64-bit complex, vector
complex quad, complex quadn imaginary quad, imaginary quadn | 128-bit complex, vector
float32m | n x m matrix of 32-bit floats
doubles32m | n x m matrix of 64-bit floats

The Khronos Group is an industry consortium creating open standards for the authoring and acceleration of parallel computing, graphics and dynamic media on a wide variety of platforms and devices. See www.khronos.org to learn more about the Khronos Group.

OpenCL is a trademark of Apple Inc. and is used under license by Khronos.