

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.

[n.n.n] refers to the section in the API Specification available at www.khronos.org/opencv.

The OpenCL Runtime

Command Queues [5.1]

```
cl_command_queue clCreateCommandQueue (
    cl_context context, cl_device_id device,
    cl_command_queue_properties properties,
    cl_int *errcode_ret)
```

properties: CL_QUEUE_PROFILING_ENABLE,
CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE

```
cl_int clRetainCommandQueue (
    cl_command_queue command_queue)
```

```
cl_int clReleaseCommandQueue (
    cl_command_queue command_queue)
```

```
cl_int clGetCommandQueueInfo (
    cl_command_queue command_queue,
    cl_command_queue_info param_name,
    size_t param_value_size,
    void *param_value,
    size_t *param_value_size_ret)
```

param_name: CL_QUEUE_CONTEXT,
CL_QUEUE_DEVICE,
CL_QUEUE_REFERENCE_COUNT,
CL_QUEUE_PROPERTIES

Buffer Objects

Elements of a buffer object can be a scalar or vector data type or a user-defined structure. Elements are stored sequentially and are accessed using a pointer by a kernel executing on a device. Data is stored in the same format as it is accessed by the kernel.

Create Buffer Objects [5.2.1]

```
cl_mem clCreateBuffer (cl_context context,
    cl_mem_flags flags, size_t size, void *host_ptr,
    cl_int *errcode_ret)
```

```
cl_mem clCreateSubBuffer (cl_mem buffer,
    cl_mem_flags flags,
    cl_buffer_create_type buffer_create_type,
    const void *buffer_create_info, cl_int *errcode_ret)
```

flags for clCreateBuffer and clCreateSubBuffer:
CL_MEM_READ_WRITE,
CL_MEM_{WRITE, READ}_ONLY,
CL_MEM_{USE, ALLOC, COPY}_HOST_PTR

Read, Write, Copy Buffer Objects [5.2.2]

```
cl_int clEnqueueReadBuffer (
    cl_command_queue command_queue, cl_mem buffer,
    cl_bool blocking_read, size_t offset, size_t cb,
    void *ptr, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

```
cl_int clEnqueueWriteBuffer (
    cl_command_queue command_queue, cl_mem buffer,
    cl_bool blocking_write, size_t offset, size_t cb,
    const void *ptr, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

Program Objects

Create Program Objects [5.6.1]

```
cl_program clCreateProgramWithSource (
    cl_context context, cl_uint count, const char **strings,
    const size_t *lengths, cl_int *errcode_ret)
```

```
cl_program clCreateProgramWithBinary (
    cl_context context, cl_uint num_devices,
    const cl_device_id *device_list, const size_t *lengths,
    const unsigned char **binaries, cl_int *binary_status,
    cl_int *errcode_ret)
```

```
cl_int clRetainProgram (cl_program program)
```

```
cl_int clReleaseProgram (cl_program program)
```

The OpenCL Platform Layer

The OpenCL platform layer implements platform-specific features that allow applications to query OpenCL devices, device configuration information, and to create OpenCL contexts using one or more devices.

Contexts [4.3]

```
cl_context clCreateContext (
    const cl_context_properties *properties, cl_uint num_devices,
    const cl_device_id *devices, void (CL_CALLBACK *pfn_notify)
    (const char *errinfo, const void *private_info,
    size_t cb, void *user_data),
    void *user_data, cl_int *errcode_ret)
```

properties: CL_CONTEXT_PLATFORM, CL_GL_CONTEXT_KHR,
CL_CGL_SHAREGROUP_KHR, CL_EGL_KHR, CL_GFX_DISPLAY_KHR,
CL_WGL_HDC_KHR

```
cl_context clCreateContextFromType (
    const cl_context_properties *properties,
    cl_device_type device_type, void (CL_CALLBACK *pfn_notify)
    (const char *errinfo, const void *private_info, size_t cb,
    void *user_data),
    void *user_data, cl_int *errcode_ret)
```

properties: See clCreateContext

```
cl_int clRetainContext (cl_context context)
```

```
cl_int clReleaseContext (cl_context context)
```

```
cl_int clGetContextInfo (cl_context context,
    cl_context_info param_name, size_t param_value_size,
    void *param_value, size_t *param_value_size_ret)
```

param_name: CL_CONTEXT_REFERENCE_COUNT,
CL_CONTEXT_DEVICES, CL_CONTEXT_PROPERTIES, CL_CONTEXT_NUM_DEVICES

Querying Platform Info and Devices [4.1, 4.2]

```
cl_int clGetPlatformIDs (cl_uint num_entries,
    cl_platform_id *platforms, cl_uint *num_platforms)
```

```
cl_int clGetPlatformInfo (cl_platform_id platform,
    cl_platform_info param_name, size_t param_value_size,
    void *param_value, size_t *param_value_size_ret)
```

param_name: CL_PLATFORM_PROFILE, CL_PLATFORM_VERSION,
CL_PLATFORM_NAME, CL_PLATFORM_VENDOR, CL_PLATFORM_EXTENSIONS

```
cl_int clGetDeviceIDs (cl_platform_id platform,
    cl_device_type device_type, cl_uint num_entries,
    cl_device_id *devices, cl_uint *num_devices)
```

device_type: CL_DEVICE_TYPE_CPU, CL_DEVICE_TYPE_GPU,
CL_DEVICE_TYPE_ACCELERATOR, CL_DEVICE_TYPE_DEFAULT, CL_DEVICE_TYPE_ALL

```
cl_int clEnqueueReadBufferRect (
    cl_command_queue command_queue, cl_mem buffer,
    cl_bool blocking_read, const size_t buffer_origin[3],
    const size_t host_origin[3], const size_t region[3],
    size_t buffer_row_pitch, size_t buffer_slice_pitch,
    size_t host_row_pitch, size_t host_slice_pitch,
    void *ptr, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

```
cl_int clEnqueueWriteBufferRect (
    cl_command_queue command_queue, cl_mem buffer,
    cl_bool blocking_write, const size_t buffer_origin[3],
    const size_t host_origin[3], const size_t region[3],
    size_t buffer_row_pitch, size_t buffer_slice_pitch,
    size_t host_row_pitch, size_t host_slice_pitch,
    void *ptr, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

```
cl_int clEnqueueCopyBuffer (
    cl_command_queue command_queue,
    cl_mem src_buffer, cl_mem dst_buffer, size_t src_offset,
    size_t dst_offset, size_t cb,
    cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

```
cl_int clEnqueueCopyBufferRect (
    cl_command_queue command_queue,
    cl_mem src_buffer, cl_mem dst_buffer,
    const size_t src_origin[3], const size_t dst_origin[3],
    const size_t region[3], size_t src_row_pitch,
    size_t src_slice_pitch, size_t dst_row_pitch,
    size_t dst_slice_pitch, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

Build Program Executable [5.6.2]

```
cl_int clBuildProgram (cl_program program,
    cl_uint num_devices, const cl_device_id *device_list,
    const char *options, void (CL_CALLBACK *pfn_notify)
    (cl_program program, void *user_data),
    void *user_data)
```

Build Options [5.6.3]

Preprocessor: (-D processed in order listed in clBuildProgram)

-D name -D name=definition -I dir

Optimization options:

-cl-opt-disable -cl-no-signed-zeros -cl-strict-aliasing
-cl-mad-enable -cl-fast-relaxed-math
-cl-finite-math-only
-cl-unsafe-math-optimizations

```
cl_int clGetDeviceInfo (cl_device_id device,
    cl_device_info param_name, size_t param_value_size,
    void *param_value, size_t *param_value_size_ret)
```

param_name: CL_DEVICE_TYPE,
CL_DEVICE_VENDOR_ID,
CL_DEVICE_MAX_COMPUTE_UNITS,
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS, CL_DEVICE_MAX_WORK_GROUP_SIZE,
CL_DEVICE_NATIVE_VECTOR_WIDTH_CHAR,
CL_DEVICE_NATIVE_VECTOR_WIDTH_SHORT,
CL_DEVICE_NATIVE_VECTOR_WIDTH_INT,
CL_DEVICE_NATIVE_VECTOR_WIDTH_LONG,
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT,
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE,
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF,
CL_DEVICE_MAX_CLOCK_FREQUENCY,
CL_DEVICE_ADDRESS_BITS,
CL_DEVICE_MAX_MEM_ALLOC_SIZE,
CL_DEVICE_IMAGE_SUPPORT,
CL_DEVICE_MAX_READ_WRITE_IMAGE_ARGS,
CL_DEVICE_IMAGE2D_MAX_WIDTH_HEIGHT,
CL_DEVICE_IMAGE3D_MAX_WIDTH_HEIGHT_DEPTH,
CL_DEVICE_MAX_SAMPLERS,
CL_DEVICE_MAX_PARAMETER_SIZE,
CL_DEVICE_MEM_BASE_ADDR_ALIGN,
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE,
CL_DEVICE_SINGLE_FP_CONFIG,
CL_DEVICE_GLOBAL_MEM_CACHE_TYPE_SIZE,
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE,
CL_DEVICE_GLOBAL_MEM_SIZE,
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE_ARGS,
CL_DEVICE_LOCAL_MEM_TYPE_SIZE,
CL_DEVICE_ERROR_CORRECTION_SUPPORT,
CL_DEVICE_PROFILING_TIMER_RESOLUTION,
CL_DEVICE_ENDIAN_LITTLE,
CL_DEVICE_AVAILABLE,
CL_DEVICE_COMPILER_AVAILABLE,
CL_DEVICE_EXECUTION_CAPABILITIES,
CL_DEVICE_QUEUE_PROPERTIES,
CL_DEVICE_NAME_VENDOR_PROFILE_EXTENSIONS,
CL_DEVICE_HOST_UNIFIED_MEMORY,
CL_DEVICE_OPENCL_C_VERSION,
CL_DEVICE_VERSION,
CL_DRIVER_VERSION, CL_DEVICE_PLATFORM

Map Buffer Objects [5.2.2]

```
void * clEnqueueMapBuffer (
    cl_command_queue command_queue, cl_mem buffer,
    cl_bool blocking_map, cl_map_flags map_flags,
    size_t offset, size_t cb, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event,
    cl_int *errcode_ret)
```

Map Buffer Objects [5.4.1-2]

```
cl_int clRetainMemObject (cl_mem memobj)
```

```
cl_int clReleaseMemObject (cl_mem memobj)
```

```
cl_int clSetMemObjectDestructorCallback (
    cl_mem memobj, void (CL_CALLBACK *pfn_notify)
    (cl_mem memobj, void *user_data),
    void *user_data)
```

```
cl_int clEnqueueUnmapMemObject (
    cl_command_queue command_queue, cl_mem memobj,
    void *mapped_ptr, cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

Query Buffer Object [5.4.3]

```
cl_int clGetMemObjectInfo (cl_mem memobj,
    cl_mem_info param_name, size_t param_value_size,
    void *param_value, size_t *param_value_size_ret)
```

param_name: CL_MEM_TYPE, CL_MEM_FLAGS, CL_MEM_SIZE, CL_MEM_HOST_PTR,
CL_MEM_MAP_REFERENCE_COUNT, CL_MEM_OFFSET,
CL_MEM_CONTEXT, CL_MEM_ASSOCIATED_MEMOBJECT

Math Intrinsics:

-cl-single-precision-constant -cl-denorms-are-zero

Warning request/suppress:

-w -Werror

Control OpenCL C language version:

-cl-std=CL1.1 // OpenCL 1.1 specification.

Query Program Objects [5.6.5]

```
cl_int clGetProgramInfo (cl_program program,
    cl_program_info param_name, size_t param_value_size,
    void *param_value, size_t *param_value_size_ret)
```

param_name: CL_PROGRAM_REFERENCE_COUNT,
CL_PROGRAM_CONTEXT, CL_PROGRAM_NUM_DEVICES, CL_PROGRAM_DEVICES,
CL_PROGRAM_SOURCE, CL_PROGRAM_BINARIES

(Program Objects Continue >)

www.khronos.org/opengl

Preprocessor Directives & Macros [6.9]

#pragma OPENCL FP_CONTRACT *on-off-switch*
on-off-switch: ON, OFF, DEFAULT

__FILE__	Current source file
__LINE__	Integer line number
__OPENCL_VERSION__	Integer version number
__CL_VERSION_1_0__	Substitutes integer 100 for version 1.0
__CL_VERSION_1_1__	Substitutes integer 110 for version 1.1
__ENDIAN_LITTLE__	1 if device is little endian
__kernel_exec(X, typen)	Same as: __kernel__attribute__((work_group_size_hint(X, 1, 1)))\n__attribute__((vec_type_hint(typen)))
__IMAGE_SUPPORT__	1 if images are supported
__FAST_RELAXED_MATH__	1 if -cl-fast-relaxed-math optimization option is specified

Math Constants [6.11.2]

The values of the following symbolic constants are type float and are accurate within the precision of a single precision floating-point number.

MAXFLOAT	Value of max. non-infinite single-precision floating-point number.
HUGE_VALF	Positive float expression, evaluates to +infinity. Used as error value.

HUGE_VAL	Positive double expression, evals. to +infinity. Used as error value. OPTIONAL	M_LN2_F	Value of loge2
INFINITY	Constant float expression, positive or unsigned infinity.	M_LN10_F	Value of loge10
NAN	Constant float expression, quiet NaN.	M_PI_F	Value of π
M_E_F	Value of e	M_PI_2_F	Value of $\pi / 2$
M_LOG2E_F	Value of log2e	M_PI_4_F	Value of $\pi / 4$
M_LOG10E_F	Value of log10e	M_1_PI_F	Value of $1 / \pi$
		M_2_PI_F	Value of $2 / \pi$
		M_2_SQRTPI_F	Value of $2 / \sqrt{\pi}$
		M_SQRT2_F	Value of $\sqrt{2}$
		M_SQRT1_2_F	Value of $1 / \sqrt{2}$

Work-Item Built-in Functions [6.11.1] *D* is dimension index.

uint get_work_dim ()	Num. of dimensions in use	size_t get_local_id (uint <i>D</i>)	Local work-item ID
size_t get_global_size (uint <i>D</i>)	Num. of global work-items	size_t get_num_groups (uint <i>D</i>)	Num. of work-groups
size_t get_global_id (uint <i>D</i>)	Global work-item ID value	size_t get_group_id (uint <i>D</i>)	Returns the work-group ID
size_t get_local_size (uint <i>D</i>)	Num. of local work-items	size_t get_global_offset (uint <i>D</i>)	Returns global offset

Integer Built-in Functions [6.11.3]

T is type char, charn, uchar, uchar_n, short, shortn, ushort, ushortn, int, intn, uint, uintn, long, longn, ulong, or ulongn. *U* is the unsigned version of *T*. *S* is the scalar version of *T*.

U abs (<i>Tx</i>)	<i>x</i>
U abs_diff (<i>Tx</i> , <i>Ty</i>)	<i>x</i> - <i>y</i> without modulo overflow
T add_sat (<i>Tx</i> , <i>Ty</i>)	<i>x</i> + <i>y</i> and saturates the result
T hadd (<i>Tx</i> , <i>Ty</i>)	(<i>x</i> + <i>y</i>) >> 1 without mod. overflow
T rhadd (<i>Tx</i> , <i>Ty</i>)	(<i>x</i> + <i>y</i> + 1) >> 1
T clz (<i>Tx</i>)	Number of leading 0-bits in <i>x</i>
T clamp (<i>Tx</i> , <i>T min</i> , <i>T max</i>)	min(max(<i>x</i> , <i>minval</i>), <i>maxval</i>)
T clamp (<i>Tx</i> , <i>S min</i> , <i>S max</i>)	
T mad_hi (<i>Ta</i> , <i>Tb</i> , <i>Tc</i>)	mul_hi(<i>a</i> , <i>b</i>) + <i>c</i>
T mad_sat (<i>Ta</i> , <i>Tb</i> , <i>Tc</i>)	<i>a</i> * <i>b</i> + <i>c</i> and saturates the result
T max (<i>Tx</i> , <i>Ty</i>)	<i>y</i> if <i>x</i> < <i>y</i> , otherwise it returns <i>x</i>
T max (<i>Tx</i> , <i>Sy</i>)	
T min (<i>Tx</i> , <i>Ty</i>)	<i>y</i> if <i>y</i> < <i>x</i> , otherwise it returns <i>x</i>
T min (<i>Tx</i> , <i>Sy</i>)	<i>y</i> if <i>y</i> < <i>x</i> , otherwise it returns <i>x</i>
T mul_hi (<i>Tx</i> , <i>Ty</i>)	high half of the product of <i>x</i> and <i>y</i>
T rotate (<i>Tv</i> , <i>Ti</i>)	result[<i>indx</i>] = <i>v</i> [<i>indx</i>] << <i>i</i> [<i>indx</i>]

T sub_sat (<i>Tx</i> , <i>Ty</i>)	<i>x</i> - <i>y</i> and saturates the result
For upsample, scalar types are permitted for the vector types below.	
shortn upsample (charn <i>hi</i> , uchar <i>lo</i>)	result[<i>i</i>] = ((short) <i>hi</i> [<i>i</i>] << 8) <i>lo</i> [<i>i</i>]
ushortn upsample (uchar <i>hi</i> , uchar <i>lo</i>)	result[<i>i</i>] = ((ushort) <i>hi</i> [<i>i</i>] << 8) <i>lo</i> [<i>i</i>]
intn upsample (shortn <i>hi</i> , ushortn <i>lo</i>)	result[<i>i</i>] = ((int) <i>hi</i> [<i>i</i>] << 16) <i>lo</i> [<i>i</i>]
uintn upsample (ushortn <i>hi</i> , ushortn <i>lo</i>)	result[<i>i</i>] = ((uint) <i>hi</i> [<i>i</i>] << 16) <i>lo</i> [<i>i</i>]
longn upsample (intn <i>hi</i> , uintn <i>lo</i>)	result[<i>i</i>] = ((long) <i>hi</i> [<i>i</i>] << 32) <i>lo</i> [<i>i</i>]
ulongn upsample (uintn <i>hi</i> , uintn <i>lo</i>)	result[<i>i</i>] = ((ulong) <i>hi</i> [<i>i</i>] << 32) <i>lo</i> [<i>i</i>]

The following fast integer functions optimize the performance of kernels. In these functions, *T* is type int, int2, int3, int4, int8, int16, uint, uint2, uint4, uint8 or uint16.

T mad24 (<i>Ta</i> , <i>Tb</i> , <i>Tc</i>)	Multiply 24-bit int. values <i>a</i> , <i>b</i> , add 32-bit int. result to 32-bit int. <i>c</i>
T mul24 (<i>Ta</i> , <i>Tb</i>)	Multiply 24-bit int. values <i>a</i> and <i>b</i>

Common Built-in Functions [6.11.4]

T is type float or floatn (or optionally double, double_n, or halfn). Optional extensions enable double, double_n, and halfn types.

T clamp (<i>Tx</i> , <i>T min</i> , <i>T max</i>) floatn clamp (floatn <i>x</i> , float <i>min</i> , float <i>max</i>) doublen clamp (doublen <i>x</i> , double <i>min</i> , double <i>max</i>) halfn clamp (halfn <i>x</i> , half <i>min</i> , half <i>max</i>)	Clamp <i>x</i> to range given by <i>min</i> , <i>max</i>
T degrees (<i>T radians</i>)	<i>radians</i> to <i>degrees</i>
T max (<i>Tx</i> , <i>Ty</i>) floatn max (floatn <i>x</i> , float <i>y</i>) doublen max (doublen <i>x</i> , double <i>y</i>) halfn max (halfn <i>x</i> , half <i>y</i>)	Max of <i>x</i> and <i>y</i>
T min (<i>Tx</i> , <i>Ty</i>) floatn min (floatn <i>x</i> , float <i>y</i>) doublen min (doublen <i>x</i> , double <i>y</i>) halfn min (halfn <i>x</i> , half <i>y</i>)	Min of <i>x</i> and <i>y</i>
T mix (<i>Tx</i> , <i>Ty</i> , <i>To</i>) floatn mix (floatn <i>x</i> , float <i>y</i> , float <i>a</i>) doublen mix (doublen <i>x</i> , double <i>y</i> , double <i>a</i>) halfn mix (halfn <i>x</i> , half <i>y</i> , half <i>a</i>)	Linear blend of <i>x</i> and <i>y</i>
T radians (<i>T degrees</i>)	<i>degrees</i> to <i>radians</i>
T step (<i>T edge</i> , <i>Tx</i>) floatn step (float <i>edge</i> , floatn <i>x</i>) doublen step (double <i>edge</i> , doublen <i>x</i>) halfn step (half <i>edge</i> , halfn <i>x</i>)	0.0 if <i>x</i> < <i>edge</i> , else 1.0
T smoothstep (<i>T edge0</i> , <i>T edge1</i> , <i>Tx</i>) floatn smoothstep (float <i>edge0</i> , float <i>edge1</i> , floatn <i>x</i>) doublen smoothstep (double <i>edge0</i> , double <i>edge1</i> , doublen <i>x</i>) halfn smoothstep (half <i>edge0</i> , half <i>edge1</i> , halfn <i>x</i>)	Step and interpolate
T sign (<i>Tx</i>)	Sign of <i>x</i>

Math Built-in Functions [6.11.2]

T is type float or floatn (or optionally double, double_n, or halfn). intn, uintn, and ulongn must be scalar when *T* is scalar. *Q* is qualifier __global, __local, or __private. **HN** indicates that Half and Native variants are available by prepending "half_" or "native_" to function name. Prototypes shown in purple are half_ and native_ only. Optional extensions enable double, double_n, half, and halfn types.

T acos (<i>T</i>)	Arc cosine
T acosh (<i>T</i>)	Inverse hyperbolic cosine
T acospi (<i>Tx</i>)	acos (<i>x</i>) / π
T asin (<i>T</i>)	Arc sine
T asinh (<i>T</i>)	Inverse hyperbolic sine
T asinpi (<i>Tx</i>)	asin (<i>x</i>) / π
T atan (<i>Ty_over_x</i>)	Arc tangent
T atan2 (<i>Ty</i> , <i>Tx</i>)	Arc tangent of <i>y</i> / <i>x</i>
T atanh (<i>T</i>)	Hyperbolic arc tangent
T atanpi (<i>Tx</i>)	atan (<i>x</i>) / π
T atan2pi (<i>Tx</i> , <i>Ty</i>)	atan2 (<i>x</i> , <i>y</i>) / π
T cbrt (<i>T</i>)	Cube root
T ceil (<i>T</i>)	Round to integer toward + infinity
T copysign (<i>Tx</i> , <i>Ty</i>)	<i>x</i> with sign changed to sign of <i>y</i>
T cos (<i>T</i>) HN	Cosine
T cosh (<i>T</i>)	Hyperbolic cosine
T cospi (<i>Tx</i>)	cos (π <i>x</i>)
T half_divide (<i>Tx</i> , <i>Ty</i>)	<i>x</i> / <i>y</i> (<i>T</i> may be float or floatn)
T native_divide (<i>Tx</i> , <i>Ty</i>)	
T erfc (<i>T</i>)	Complementary error function
T erf (<i>T</i>)	Calculates error function of <i>T</i>
T exp (<i>Tx</i>) HN	Exponential base e
T exp2 (<i>T</i>) HN	Exponential base 2
T exp10 (<i>T</i>) HN	Exponential base 10

T expm1 (<i>Tx</i>)	$e^x - 1.0$
T fabs (<i>T</i>)	Absolute value
T fdim (<i>Tx</i> , <i>Ty</i>)	"Positive difference" between <i>x</i> and <i>y</i>
T floor (<i>T</i>)	Round to integer toward - infinity
T fma (<i>Ta</i> , <i>Tb</i> , <i>Tc</i>)	Multiply and add, then round
T fmax (<i>Tx</i> , <i>Ty</i>) halfn fmax (halfn <i>x</i> , half <i>y</i>) floatn fmax (floatn <i>x</i> , float <i>y</i>) doublen fmax (doublen <i>x</i> , double <i>y</i>)	Return <i>y</i> if <i>x</i> < <i>y</i> , otherwise it returns <i>x</i>
T fmin (<i>Tx</i> , <i>Ty</i>) halfn fmin (halfn <i>x</i> , half <i>y</i>) floatn fmin (floatn <i>x</i> , float <i>y</i>) doublen fmin (doublen <i>x</i> , double <i>y</i>)	Return <i>y</i> if <i>y</i> < <i>x</i> , otherwise it returns <i>x</i>
T fmod (<i>Tx</i> , <i>Ty</i>)	Modulus. Returns <i>x</i> - <i>y</i> * trunc (<i>x</i> / <i>y</i>)
T fract (<i>Tx</i> , <i>Q T *iptr</i>)	Fractional value in <i>x</i>
T frexp (<i>Tx</i> , <i>Q intrn *exp</i>)	Extract mantissa and exponent
T hypot (<i>Tx</i> , <i>Ty</i>)	Square root of <i>x</i> ² + <i>y</i> ²
intn ilogb (<i>Tx</i>)	Return exponent as an integer value
T ldxexp (<i>Tx</i> , intrn <i>n</i>) T ldxexp (<i>Tx</i> , int <i>n</i>)	<i>x</i> * 2 ^{<i>n</i>}
T lgamma (<i>Tx</i>) T lgamma_r (<i>Tx</i> , <i>Q intrn *signp</i>)	Log gamma function
T log (<i>T</i>) HN	Natural logarithm
T log2 (<i>T</i>) HN	Base 2 logarithm
T log10 (<i>T</i>) HN	Base 10 logarithm
T log1p (<i>Tx</i>)	ln (1.0 + <i>x</i>)
T logb (<i>Tx</i>)	Exponent of <i>x</i>
T mad (<i>Ta</i> , <i>Tb</i> , <i>Tc</i>)	Approximates <i>a</i> * <i>b</i> + <i>c</i>
T maxmag (<i>Tx</i> , <i>Ty</i>)	Maximum magnitude of <i>x</i> and <i>y</i>

T minmag (<i>Tx</i> , <i>Ty</i>)	Minimum magnitude of <i>x</i> and <i>y</i>
T modf (<i>Tx</i> , <i>Q T *iptr</i>)	Decompose a floating-point number
float nan (uintn <i>nancode</i>) floatn nan (uintn <i>nancode</i>) halfn nan (ushortn <i>nancode</i>) doublen nan (ulongn <i>nancode</i>)	Quiet NaN
T nextafter (<i>Tx</i> , <i>Ty</i>)	Next representable floating-point value following <i>x</i> in the direction of <i>y</i>
T pow (<i>Tx</i> , <i>Ty</i>)	Compute <i>x</i> to the power of <i>y</i> (<i>x</i> ^{<i>y</i>})
T pown (<i>Tx</i> , intrn <i>y</i>)	Compute <i>x</i> ^{<i>y</i>} , where <i>y</i> is an integer
T powr (<i>Tx</i> , <i>Ty</i>) HN	Compute <i>x</i> ^{<i>y</i>} , where <i>x</i> is >= 0
T half_recip (<i>Tx</i>) T native_recip (<i>Tx</i>)	1 / <i>x</i> (<i>T</i> may be float or floatn)
T remainder (<i>Tx</i> , <i>Ty</i>)	Floating point remainder
T remquo (<i>Tx</i> , <i>Ty</i> , Q intrn *quo)	Floating point remainder and quotient
T rint (<i>T</i>)	Round to nearest even integer
T rootn (<i>Tx</i> , intrn <i>y</i>)	Compute <i>x</i> to the power of 1/ <i>y</i>
T round (<i>Tx</i>)	Integral value nearest to <i>x</i> rounding
T rsqrt (<i>T</i>) HN	Inverse square root
T sin (<i>T</i>) HN	Sine
T sincos (<i>Tx</i> , <i>Q T *cosval</i>)	Sine and cosine of <i>x</i>
T sinh (<i>T</i>)	Hyperbolic sine
T sinpi (<i>Tx</i>)	sin (π <i>x</i>)
T sqrt (<i>T</i>) HN	Square root
T tan (<i>T</i>) HN	Tangent
T tanh (<i>T</i>)	Hyperbolic tangent
T tanpi (<i>Tx</i>)	tan (π <i>x</i>)
T tgamma (<i>T</i>)	Gamma function
T trunc (<i>T</i>)	Round to integer toward zero

Geometric Built-in Functions [6.11.5]

Vector types may have 2, 3, or 4 components. **Optional extensions enable double, doublen, and halfn types.**

float dot (float <i>p0</i> , float <i>p1</i>) float dot (floatn <i>p0</i> , floatn <i>p1</i>) double dot (double <i>p0</i> , double <i>p1</i>) double dot (doublen <i>p0</i> , doublen <i>p1</i>) half dot (half <i>p0</i> , half <i>p1</i>) half dot (halfn <i>p0</i> , halfn <i>p1</i>)	Dot product
float[3,4] cross (float[3,4] <i>p0</i> , float[3,4] <i>p1</i>) double[3,4] cross (double[3,4] <i>p0</i> , double[3,4] <i>p1</i>) half[3,4] cross (half[3,4] <i>p0</i> , half[3,4] <i>p1</i>)	Cross product

float distance (float <i>p0</i> , float <i>p1</i>) float distance (floatn <i>p0</i> , floatn <i>p1</i>) double distance (double <i>p0</i> , double <i>p1</i>) double distance (doublen <i>p0</i> , doublen <i>p1</i>) half distance (half <i>p0</i> , half <i>p1</i>) half distance (halfn <i>p0</i> , halfn <i>p1</i>)	Vector distance
float length (float <i>p</i>) float length (floatn <i>p</i>) double length (double <i>p</i>) double length (doublen <i>p</i>) half length (half <i>p</i>) half length (halfn <i>p</i>)	Vector length

float normalize (float <i>p</i>) floatn normalize (floatn <i>p</i>) double normalize (double <i>p</i>) doublen normalize (doublen <i>p</i>) half normalize (half <i>p</i>) halfn normalize (halfn <i>p</i>)	Normal vector length 1
float fast_distance (float <i>p0</i> , float <i>p1</i>) float fast_distance (floatn <i>p0</i> , floatn <i>p1</i>)	Vector distance
float fast_length (float <i>p</i>) float fast_length (floatn <i>p</i>)	Vector length
float fast_normalize (float <i>p</i>) floatn fast_normalize (floatn <i>p</i>)	Normal vector length 1

Relational Built-in Functions [6.11.6]

T is type float, floatn, char, charn, uchar, uchar, short, shortn, ushort, ushortn, int, intrn, uint, uintn, long, longn, ulong, or ulongn (and optionally double, doublen). *S* is type char, charn, short, shortn, int, intrn, long, or longn. *U* is type uchar, uchar, ushort, ushortn, uint, uintn, ulong, or ulongn. **Optional extensions enable double, doublen, and halfn types.**

int isequal (float <i>x</i> , float <i>y</i>) intrn isequal (floatn <i>x</i> , floatn <i>y</i>) int isequal (double <i>x</i> , double <i>y</i>) longn isequal (doublen <i>x</i> , doublen <i>y</i>) int isequal (half <i>x</i> , half <i>y</i>) shortn isequal (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x == y$
int isnotequal (float <i>x</i> , float <i>y</i>) intrn isnotequal (floatn <i>x</i> , floatn <i>y</i>) int isnotequal (double <i>x</i> , double <i>y</i>) longn isnotequal (doublen <i>x</i> , doublen <i>y</i>) int isnotequal (half <i>x</i> , half <i>y</i>) shortn isnotequal (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x != y$
int isgreater (float <i>x</i> , float <i>y</i>) intrn isgreater (floatn <i>x</i> , floatn <i>y</i>) int isgreater (double <i>x</i> , double <i>y</i>) longn isgreater (doublen <i>x</i> , doublen <i>y</i>) int isgreater (half <i>x</i> , half <i>y</i>) shortn isgreater (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x > y$
int isgreaterequal (float <i>x</i> , float <i>y</i>) intrn isgreaterequal (floatn <i>x</i> , floatn <i>y</i>) int isgreaterequal (double <i>x</i> , double <i>y</i>) longn isgreaterequal (doublen <i>x</i> , doublen <i>y</i>) int isgreaterequal (half <i>x</i> , half <i>y</i>) shortn isgreaterequal (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x >= y$
int isless (float <i>x</i> , float <i>y</i>) intrn isless (floatn <i>x</i> , floatn <i>y</i>) int isless (double <i>x</i> , double <i>y</i>) longn isless (doublen <i>x</i> , doublen <i>y</i>) int isless (half <i>x</i> , half <i>y</i>) shortn isless (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x < y$
int islessequal (float <i>x</i> , float <i>y</i>) intrn islessequal (floatn <i>x</i> , floatn <i>y</i>) int islessequal (double <i>x</i> , double <i>y</i>) longn islessequal (doublen <i>x</i> , doublen <i>y</i>) int islessequal (half <i>x</i> , half <i>y</i>) shortn islessequal (halfn <i>x</i> , halfn <i>y</i>)	Compare of $x <= y$
int islessgreater (float <i>x</i> , float <i>y</i>) intrn islessgreater (floatn <i>x</i> , floatn <i>y</i>) int islessgreater (double <i>x</i> , double <i>y</i>) longn islessgreater (doublen <i>x</i> , doublen <i>y</i>) int islessgreater (half <i>x</i> , half <i>y</i>) shortn islessgreater (halfn <i>x</i> , halfn <i>y</i>)	Compare of $(x < y) \vee (x > y)$
int isfinite (float) intrn isfinite (floatn) int isfinite (double) longn isfinite (doublen) int isfinite (half) shortn isfinite (halfn)	Test for finite value

int isninf (float) intrn isninf (floatn) int isninf (double) longn isninf (doublen) int isninf (half) shortn isninf (halfn)	Test for +ve or -ve infinity
int isnan (float) intrn isnan (floatn) int isnan (double) longn isnan (doublen) int isnan (half) shortn isnan (halfn)	Test for a NaN
int isnormal (float) intrn isnormal (floatn) int isnormal (double) longn isnormal (doublen) int isnormal (half) shortn isnormal (halfn)	Test for a normal value
int isordered (float <i>x</i> , float <i>y</i>) intrn isordered (floatn <i>x</i> , floatn <i>y</i>) int isordered (double <i>x</i> , double <i>y</i>) longn isordered (doublen <i>x</i> , doublen <i>y</i>) int isordered (half <i>x</i> , half <i>y</i>) shortn isordered (halfn <i>x</i> , halfn <i>y</i>)	Test if arguments are ordered
int isunordered (float <i>x</i> , float <i>y</i>) intrn isunordered (floatn <i>x</i> , floatn <i>y</i>) int isunordered (double <i>x</i> , double <i>y</i>) longn isunordered (doublen <i>x</i> , doublen <i>y</i>) int isunordered (half <i>x</i> , half <i>y</i>) shortn isunordered (halfn <i>x</i> , halfn <i>y</i>)	Test if arguments are unordered
int signbit (float) intrn signbit (floatn) int signbit (double) longn signbit (doublen) int signbit (half) shortn signbit (halfn)	Test for sign bit
int any (<i>S</i> <i>x</i>)	1 if MSB in any component of <i>x</i> is set; else 0
int all (<i>S</i> <i>x</i>)	1 if MSB in all components of <i>x</i> are set; else 0
<i>T</i> bitselect (<i>T</i> <i>a</i> , <i>T</i> <i>b</i> , <i>T</i> <i>c</i>) halfn bitselect (halfn <i>a</i> , halfn <i>b</i> , halfn <i>c</i>) doublen bitselect (doublen <i>a</i> , doublen <i>b</i> , doublen <i>c</i>)	Each bit of result is corresponding bit of <i>a</i> if corresponding bit of <i>c</i> is 0
<i>T</i> select (<i>T</i> <i>a</i> , <i>T</i> <i>b</i> , <i>S</i> <i>c</i>) <i>T</i> select (<i>T</i> <i>a</i> , <i>T</i> <i>b</i> , <i>U</i> <i>c</i>) doublen select (doublen, doublen, longn) doublen select (doublen, doublen, ulongn) halfn select (halfn, halfn, shortn) halfn select (halfn, halfn, ushortn)	For each component of a vector type, result[<i>i</i>] = if MSB of <i>c</i> [<i>i</i>] is set ? <i>b</i> [<i>i</i>] : <i>a</i> [<i>i</i>] For scalar type, result = <i>c</i> ? <i>b</i> : <i>a</i>

Vector Data Load/Store Functions [6.11.7]

Q is an Address Space Qualifier listed in 6.5 unless otherwise noted. *R* defaults to the current rounding mode, or is one of the Rounding Modes listed in 6.2.3.2. *T* is type char, uchar, short, ushort, int, uint, long, ulong, half, or float (or optionally double). *Tn* refers to the vector form of type *T*. **Optional extensions enable the double, doublen, half, and halfn types.**

<i>Tn</i> vloadn (size_t <i>offset</i> , const <i>Q</i> <i>T</i> * <i>p</i>)	Read vector data from memory
void vstoren (<i>Tn</i> <i>data</i> , size_t <i>offset</i> , <i>Q</i> <i>T</i> * <i>p</i>)	Write vector data to memory (<i>Q</i> in this function cannot be __constant)
float vload_half (size_t <i>offset</i> , const <i>Q</i> half * <i>p</i>)	Read a half from memory
floatn vload_halfn (size_t <i>offset</i> , const <i>Q</i> half * <i>p</i>)	Read multiple halves from memory
void vstore_half (float <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_half_R (float <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_half (double <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_half_R (double <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>)	Write a half to memory (<i>Q</i> in this function cannot be __constant)
void vstore_halfn (floatn <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_halfn_R (floatn <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_halfn (doublen <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstore_halfn_R (doublen <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>)	Write a half vector to memory (<i>Q</i> in this function cannot be __constant)
floatn vloada_halfn (size_t <i>offset</i> , const <i>Q</i> half * <i>p</i>)	sizeof (floatn) bytes of data read from location (<i>p</i> + (<i>offset</i> * <i>n</i>))
void vstorea_halfn (floatn <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstorea_halfn_R (floatn <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstorea_halfn (doublen <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>) void vstorea_halfn_R (doublen <i>data</i> , size_t <i>offset</i> , <i>Q</i> half * <i>p</i>)	Write a half vector to vector-aligned memory (<i>Q</i> in this function cannot be __constant)

Async Copies and Prefetch Functions [6.11.10]

T is type char, charn, uchar, uchar, short, shortn, ushort, ushortn, int, intrn, uint, uintn, long, longn, ulong, ulongn, float, floatn, and optionally halfn double, doublen. **Optional extensions enable the halfn, double, and doublen types.**

event_t async_work_group_copy (_local <i>T</i> * <i>dst</i> , const _global <i>T</i> * <i>src</i> , size_t <i>num_gentypes</i> , event_t <i>event</i>)	Copies <i>num_gentypes</i> <i>T</i> elements from <i>src</i> to <i>dst</i>
event_t async_work_group_copy (_global <i>T</i> * <i>dst</i> , const _local <i>T</i> * <i>src</i> , size_t <i>num_gentypes</i> , event_t <i>event</i>)	
event_t async_work_group_strided_copy (_local <i>T</i> * <i>dst</i> , const _global <i>T</i> * <i>src</i> , size_t <i>num_gentypes</i> , size_t <i>src_stride</i> , event_t <i>event</i>)	Copies <i>num_gentypes</i> <i>T</i> elements from <i>src</i> to <i>dst</i>
event_t async_work_group_strided_copy (_global <i>T</i> * <i>dst</i> , const _local <i>T</i> * <i>src</i> , size_t <i>num_gentypes</i> , size_t <i>dst_stride</i> , event_t <i>event</i>)	
void wait_group_events (int <i>num_events</i> , event_t * <i>event_list</i>)	Wait for events that identify the async_work_group_copy operations to complete
void prefetch (const _global <i>T</i> * <i>p</i> , size_t <i>num_gentypes</i>)	Prefetch <i>num_gentypes</i> * sizeof(<i>T</i>) bytes into the global cache

Atomic Functions [6.11.11, 9.4]

T is type int or unsigned int. *T* may also be type float for atomic_xchg, and type long or ulong for extended 64-bit atomic functions. *Q* is volatile __global or volatile __local, except *Q* must be volatile __global for atomic_xchg when *T* is float.

The built-in atomic functions for 32-bit values begin with atomic_ while the extended 64-bit atomic functions begin with atom_. For example:

Built-in atomic function atomic_add ()	Extended atomic function atom_add ()
-------------------------------------------	-----------------------------------------

Extended 64-bit atomic functions are enabled by the following pragma; *extension-name* is one of cl_khr_int64_ {base, extended}_atomics:

```
#pragma OPENCL EXTENSION extension-name : enable
```

<i>T</i> atomic_add (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, add, and store
<i>T</i> atomic_sub (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, subtract, and store
<i>T</i> atomic_xchg (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, swap, and store
<i>T</i> atomic_inc (<i>Q</i> <i>T</i> * <i>p</i>)	Read, increment, and store
<i>T</i> atomic_dec (<i>Q</i> <i>T</i> * <i>p</i>)	Read, decrement, and store
<i>T</i> atomic_cmpxchg (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>cmp</i> , <i>T</i> <i>val</i>)	Read and store (* <i>p</i> == <i>cmp</i>) ? <i>val</i> : * <i>p</i>
<i>T</i> atomic_min (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, store min(* <i>p</i> , <i>val</i>)
<i>T</i> atomic_max (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, store max(* <i>p</i> , <i>val</i>)
<i>T</i> atomic_and (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, store (* <i>p</i> & <i>val</i>)
<i>T</i> atomic_or (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, store (* <i>p</i> <i>val</i>)
<i>T</i> atomic_xor (<i>Q</i> <i>T</i> * <i>p</i> , <i>T</i> <i>val</i>)	Read, store (* <i>p</i> ^ <i>val</i>)

Miscellaneous Vector Built-In Functions [6.11.12]

T_n and T_m mean the 2,4,8, or 16-component vectors of char, uchar, short, ushort, half, int, uint, long, ulong, float, double. Un means the built-in unsigned integer data types. For `vec_step()`, T_n also includes char3, uchar3, short3, ushort3, half3, int3, uint3, long3, ulong3, float3, and double3. Half and double types are enabled by `cl_khr_fp16` and `cl_khr_fp64` respectively.

<code>int vec_step (Tn a)</code> <code>int vec_step (typename)</code>	Takes a built-in scalar or vector data type argument and returns an integer value representing the number of elements in the scalar or vector.
<code>Tn shuffle (Tm x, Un mask)</code> <code>Tn shuffle2 (Tm x, Tm y, Un mask)</code>	Construct permutation of elements from one or two input vectors, return a vector with same element type as input & length that is the same as the shuffle mask.

OpenCL Graphics: Following is a subset of the OpenCL API specification that pertains to graphics.

Image Read and Write Built-in Functions [6.11.13, 9.5, 9.6.8]

The built-in functions defined in this section can only be used with image memory objects created with `clCreateImage2D` or `clCreateImage3D`. `sampler` specifies the addressing and filtering mode to use. **H** = To enable `read_imageh` and `write_imageh`, enable extension `cl_khr_fp16`. **3D** = To enable type `image3d_t` in `write_image{f, i, ui}`, enable extension `cl_khr_3d_image_writes`.

<code>float4 read_imagef (image2d_t image, sampler_t sampler, int2 coord)</code> <code>float4 read_imagef (image2d_t image, sampler_t sampler, float2 coord)</code>	Read an element from a 2D image	<code>uint4 read_imageui (image3d_t image, sampler_t sampler, int4 coord)</code> <code>uint4 read_imageui (image3d_t image, sampler_t sampler, float4 coord)</code>	Read an element from a 3D image
<code>int4 read_imagei (image2d_t image, sampler_t sampler, int2 coord)</code> <code>int4 read_imagei (image2d_t image, sampler_t sampler, float2 coord)</code>		<code>int get_image_width (image2d_t image)</code> <code>int get_image_width (image3d_t image)</code>	Image width in pixels
<code>uint4 read_imageui (image2d_t image, sampler_t sampler, int2 coord)</code> <code>uint4 read_imageui (image2d_t image, sampler_t sampler, float2 coord)</code>		<code>int get_image_height (image2d_t image)</code> <code>int get_image_height (image3d_t image)</code>	Image height in pixels
<code>half4 read_imageh (image2d_t image, sampler_t sampler, int2 coord)</code> H <code>half4 read_imageh (image2d_t image, sampler_t sampler, float2 coord)</code> H		<code>int get_image_depth (image3d_t image)</code>	Image depth in pixels
<code>void write_imagef (image2d_t image, int2 coord, float4 color)</code> <code>void write_imagei (image2d_t image, int2 coord, int4 color)</code> <code>void write_imageui (image2d_t image, int2 coord, uint4 color)</code> <code>void write_imageh (image2d_t image, int2 coord, half4 color)</code> H	Write <i>color</i> value to (x, y) location specified by <i>coord</i> in the 2D image	<code>int get_image_channel_data_type (image2d_t image)</code> <code>int get_image_channel_data_type (image3d_t image)</code>	Image channel data type
<code>float4 read_imagef (image3d_t image, sampler_t sampler, int4 coord)</code> <code>float4 read_imagef (image3d_t image, sampler_t sampler, float4 coord)</code> <code>int4 read_imagei (image3d_t image, sampler_t sampler, int4 coord)</code> <code>int4 read_imagei (image3d_t image, sampler_t sampler, float4 coord)</code>	Read an element from a 3D image	<code>int get_image_channel_order (image2d_t image)</code> <code>int get_image_channel_order (image3d_t image)</code>	Image channel order
		<code>int2 get_image_dim (image2d_t image)</code> <code>int4 get_image_dim (image3d_t image)</code>	Image width, height Image width, height, and depth
		Use this pragma to enable type <code>image3d_t</code> in <code>write_image{f, i, ui}</code> : <code>#pragma OPENCL EXTENSION cl_khr_3d_image_writes : enable</code>	Writes <i>color</i> at <i>coord</i> in the 3D image
		<code>void write_imagef (image3d_t image, int4 coord, float4 color)</code> 3D <code>void write_imagei (image3d_t image, int4 coord, int4 color)</code> 3D <code>void write_imageui (image3d_t image, int4 coord, uint4 color)</code> 3D	

Image Objects**Create Image Objects** [5.3.1]

`cl_mem clCreateImage2D (cl_context context, cl_mem_flags flags, const cl_image_format *image_format, size_t image_width, size_t image_height, size_t image_row_pitch, void *host_ptr, cl_int *errcode_ret)`
flags: (also for `clCreateImage3D`, `clGetSupportedImageFormats`)
`CL_MEM_READ_WRITE`, `CL_MEM_WRITE_READ_ONLY`, `CL_MEM_USE_ALLOC_COPY_HOST_PTR`

`cl_mem clCreateImage3D (cl_context context, cl_mem_flags flags, const cl_image_format *image_format, size_t image_width, size_t image_height, size_t image_depth, size_t image_row_pitch, size_t image_slice_pitch, void *host_ptr, cl_int *errcode_ret)`

flags: See `clCreateImage2D`

Query List of Supported Image Formats [5.3.2]

`cl_int clGetSupportedImageFormats (cl_context context, cl_mem_flags flags, cl_mem_object_type image_type, cl_uint num_entries, cl_image_format *image_formats, cl_uint *num_image_formats)`

flags: See `clCreateImage2D`

Copy Between Image, Buffer Objects [5.3.4]

`cl_int clEnqueueCopyImageToBuffer (cl_command_queue command_queue, cl_mem src_image, cl_mem dst_buffer, const size_t src_origin[3], const size_t dst_origin[3], const size_t dst_offset, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)`

`cl_int clEnqueueCopyBufferToImage (cl_command_queue command_queue, cl_mem src_buffer, cl_mem dst_image, size_t src_offset, const size_t dst_origin[3], const size_t region[3], cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)`

Map and Unmap Image Objects [5.3.5]

`void * clEnqueueMapImage (cl_command_queue command_queue, cl_mem image, cl_bool blocking_map, cl_map_flags map_flags, const size_t origin[3], const size_t region[3], size_t *image_row_pitch, size_t *image_slice_pitch, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event, cl_int *errcode_ret)`

Read, Write, Copy Image Objects [5.3.3]

`cl_int clEnqueueReadImage (cl_command_queue command_queue, cl_mem image, cl_bool blocking_read, const size_t origin[3], const size_t region[3], size_t row_pitch, size_t slice_pitch, void *ptr, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)`

`cl_int clEnqueueWriteImage (cl_command_queue command_queue, cl_mem image, cl_bool blocking_write, const size_t origin[3], const size_t region[3], size_t input_row_pitch, size_t input_slice_pitch, const void *ptr, cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)`

`cl_int clEnqueueCopyImage (cl_command_queue command_queue, cl_mem src_image, cl_mem dst_image, const size_t src_origin[3], const size_t dst_origin[3], const size_t region[3], cl_uint num_events_in_wait_list, const cl_event *event_wait_list, cl_event *event)`

Query Image Objects [5.3.6]

`cl_int clGetMemObjectInfo (cl_mem memobj, cl_mem_info param_name, size_t param_value_size, void *param_value, size_t *param_value_size_ret)`
param_name: `CL_MEM_TYPE`, `CL_MEM_FLAGS`, `CL_MEM_HOST_PTR`, `CL_MEM_MAP_REFERENCE_COUNT`, `CL_MEM_CONTEXT_OFFSET`, `CL_MEM_ASSOCIATED_MEMOBJECT`

`cl_int clGetImageInfo (cl_mem image, cl_image_info param_name, size_t param_value_size, void *param_value, size_t *param_value_size_ret)`
param_name: `CL_IMAGE_FORMAT`, `CL_IMAGE_ELEMENT_SIZE`, `CL_IMAGE_ROW_SLICE_PITCH`, `CL_IMAGE_HEIGHT_WIDTH_DEPTH`, `CL_IMAGE_D3D10_SUBRESOURCE_KHR`, `CL_MEM_D3D10_RESOURCE_KHR`

Access Qualifiers [6.6]

Apply to image `image2d_t` and `image3d_t` types to declare if the image memory object is being read or written by a kernel. The default qualifier is `__read_only`.

`__read_only`, `read_only`
`__write_only`, `write_only`

Synchronization, Explicit Mem. Fence [6.11.9-10]

flags argument is the memory address space, set to a combination of `CLK_LOCAL_MEM_FENCE` and `CLK_GLOBAL_MEM_FENCE`.

<code>void barrier (cl_mem_fence_flags flags)</code>	All work-items in a work-group must execute this before any can continue
<code>void mem_fence (cl_mem_fence_flags flags)</code>	Orders loads and stores of a work-item executing a kernel
<code>void read_mem_fence (cl_mem_fence_flags flags)</code>	Orders memory loads
<code>void write_mem_fence (cl_mem_fence_flags flags)</code>	Orders memory stores

Image Formats [5.3.1.1, 9.5]

Supported image formats: `image_channel_order` with `image_channel_data_type`.

Built-in support: [Table 5.7]

CL_RGBA: `CL_HALF_FLOAT`, `CL_FLOAT`, `CL_UNORM_INT8[16]`, `CL_SIGNED_INT8[16,32]`, `CL_UNSIGNED_INT8[16,32]`

CL_BGRA: `CL_UNORM_INT8`

Optional support: [Table 5.5]

CL_R, **CL_A**: `CL_HALF_FLOAT`, `CL_FLOAT`, `CL_UNORM_INT8[16]`, `CL_SIGNED_INT8[16,32]`, `CL_UNSIGNED_INT8[16,32]`, `CL_SNORM_INT8[16]`

CL_INTENSITY: `CL_HALF_FLOAT`, `CL_FLOAT`, `CL_UNORM_INT8[16]`, `CL_SNORM_INT8[16]`

CL_LUMINANCE: `CL_UNORM_INT8[16]`, `CL_HALF_FLOAT`, `CL_FLOAT`, `CL_SNORM_INT8[16]`

CL_RG, **CL_RA**: `CL_HALF_FLOAT`, `CL_FLOAT`, `CL_UNORM_INT8[16]`, `CL_SIGNED_INT8[16,32]`, `CL_UNSIGNED_INT8[16,32]`, `CL_SNORM_INT8[16]`

CL_RGB: `CL_UNORM_SHORT[555,565]`, `CL_UNORM_INT101010`

CL_ARGB: `CL_UNORM_INT8`, `CL_SIGNED_INT8`, `CL_UNSIGNED_INT8`, `CL_SNORM_INT8`

CL_BGRA: `CL_SIGNED_INT8`, `CL_UNSIGNED_INT8`, `CL_SNORM_INT8`

Sampler Objects [5.5]

`cl_sampler clCreateSampler (cl_context context, cl_bool normalized_coors, cl_addressing_mode addressing_mode, cl_filter_mode filter_mode, cl_int *errcode_ret)`

`cl_int clRetainSampler (cl_sampler sampler)`

`cl_int clReleaseSampler (cl_sampler sampler)`

`cl_int clGetSamplerInfo (cl_sampler sampler, cl_sampler_info param_name, size_t param_value_size, void *param_value, size_t *param_value_size_ret)`

param_name: `CL_SAMPLER_REFERENCE_COUNT`, `CL_SAMPLER_CONTEXT_FILTER_MODE`, `CL_SAMPLER_ADDRESSING_MODE`, `CL_SAMPLER_NORMALIZED_COORDS`

Sampler Declaration Fields [6.11.13.1]

The sampler can be passed as an argument to the kernel using `clSetKernelArg`, or it can be a constant variable of type `sampler_t` declared in the program source.

```
const sampler_t <sampler-name> =
    <normalized-mode> | <address-mode> | <filter-mode>
```

normalized-mode:
CLK_NORMALIZED_COORDS_{TRUE, FALSE}

address-mode:
CLK_ADDRESS_{REPEAT, CLAMP, NONE},
CLK_ADDRESS_{CLAMP_TO_EDGE, MIRRORED_REPEAT}

filter-mode:
CLK_FILTER_NEAREST, CLK_FILTER_LINEAR

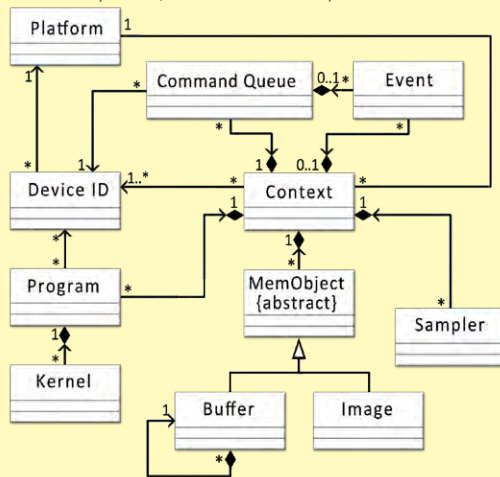
OpenCL Class Diagram [5.13]

The figure below describes the OpenCL specification as a class diagram using the Unified Modeling Language¹ (UML) notation. The diagram shows both nodes and edges which are classes and their relationships. As a simplification it shows only classes, and no attributes or operations.

Annotations

Relationships	
abstract classes	{abstract}
aggregations	◆
inheritance	△
relationship navigability	^

Cardinality	
many	*
one and only one	1
optionally one	0..1
one or more	1..*



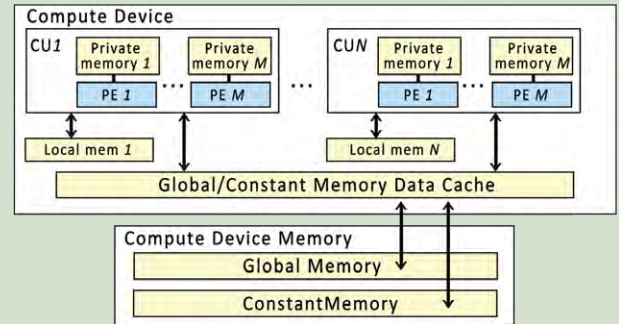
¹ Unified Modeling Language (<http://www.uml.org/>) is a trademark of Object Management Group (OMG).

OpenCL Device Architecture Diagram [3.3]

The table below shows memory regions with allocation and memory access capabilities.

	Global	Constant	Local	Private
Host	Dynamic allocation Read/Write access	Dynamic allocation Read/Write access	Dynamic allocation No access	No allocation No access
Kernel	No allocation Read/Write access	Static allocation Read-only access	Static allocation Read/Write access	Static allocation Read/Write access

This conceptual OpenCL device architecture diagram shows processing elements (PE), compute units (CU), and devices. The host is not shown.

**OpenCL/OpenGL Sharing APIs**

Creating OpenCL memory objects from OpenGL objects using `clCreateFromGLBuffer`, `clCreateFromGLTexture2D`, `clCreateFromGLTexture3D`, and `clCreateFromGLRenderbuffer` ensure that the storage of the OpenGL object will not be deleted while the corresponding OpenCL memory object exists.

CL Buffer Objects > GL Buffer Objects [9.8.2]

```
cl_mem clCreateFromGLBuffer (cl_context context,
    cl_mem_flags flags, GLuint bufobj, int *errcode_ret)
flags: CL_MEM_(READ, WRITE)_ONLY, CL_MEM_READ_WRITE
```

CL Image Objects > GL Textures [9.8.3]

```
cl_mem clCreateFromGLTexture2D (cl_context context,
    cl_mem_flags flags, GLuint texture_target,
    GLint miplevel, GLuint texture, cl_int *errcode_ret)
flags: See clCreateFromGLBuffer
texture_target: GL_TEXTURE_{2D, RECTANGLE},
    GL_TEXTURE_CUBE_MAP_POSITIVE_{X, Y, Z},
    GL_TEXTURE_CUBE_MAP_NEGATIVE_{X, Y, Z}
```

```
cl_mem clCreateFromGLTexture3D (cl_context context,
    cl_mem_flags flags, GLuint texture_target,
    GLint miplevel, GLuint texture, cl_int *errcode_ret)
flags: See clCreateFromGLBuffer
texture_target: GL_TEXTURE_3D
```

CL Image Objects > GL Renderbuffers [9.8.4]

```
cl_mem clCreateFromGLRenderbuffer (cl_context context, cl_mem_flags flags,
    GLuint renderbuffer, cl_int *errcode_ret)
flags: clCreateFromGLBuffer
```

Query Information [9.8.5]

```
cl_int clGetGLObjectInfo (cl_mem memobj,
    cl_gl_object_type *gl_object_type, GLuint *gl_object_name)
*gl_object_type returns: CL_GL_OBJECT_BUFFER,
    CL_GL_OBJECT_TEXTURE2D, TEXTURE3D,
    CL_GL_OBJECT_RENDERBUFFER
```

```
cl_int clGetGLTextureInfo (cl_mem memobj,
    cl_gl_texture_info param_name,
    size_t param_value_size, void *param_value,
    size_t *param_value_size_ret)
param_name: CL_GL_TEXTURE_TARGET,
    CL_GL_MIPMAP_LEVEL
```

Share Objects [9.8.6]

```
cl_int clEnqueueAcquireGLObjects (
    cl_command_queue command_queue,
    cl_uint num_objects, const cl_mem *mem_objects,
    cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

```
cl_int clEnqueueReleaseGLObjects (
    cl_command_queue command_queue,
    cl_uint num_objects, const cl_mem *mem_objects,
    cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list, cl_event *event)
```

CL Event Objects > GL Sync Objects [9.9]

```
cl_event clCreateEventFromGLsyncKHR (
    cl_context context, GLsync sync, cl_int *errcode_ret)
```

CL Context > GL Context, Sharegroup [9.7]

```
cl_int clGetGLContextInfoKHR (
    const cl_context_properties *properties,
    cl_gl_context_info param_name,
    size_t param_value_size, void *param_value,
    size_t *param_value_size_ret)
param_name: CL_DEVICES_FOR_GL_CONTEXT_KHR,
    CL_CURRENT_DEVICE_FOR_GL_CONTEXT_KHR
```

OpenCL/Direct3D 10 Sharing APIs [9.10]

Creating OpenCL memory objects from OpenGL objects using `clCreateFromGLBuffer`, `clCreateFromGLTexture2D`, `clCreateFromGLTexture3D`, or `clCreateFromGLRenderbuffer` ensures that the storage of that OpenGL object will not be deleted while the corresponding OpenCL memory object exists.

```
cl_int clGetDeviceIDsFromD3D10KHR (
    cl_platform_id platform,
    cl_d3d10_device_source_khr d3d_device_source,
    void *d3d_object, cl_d3d10_device_set_khr
    d3d_device_set, cl_uint num_entries,
    cl_device_id *devices, cl_uint *num_devices)
d3d_device_source: CL_D3D10_DEVICE_KHR,
    CL_D3D10_DXGI_ADAPTER_KHR
d3d_object: ID3D10Device, IDXGIAdapter
d3d_device_set: CL_ALL_DEVICES_FOR_D3D10_KHR,
    CL_PREFERRED_DEVICES_FOR_D3D10_KHR
```

```
cl_mem clCreateFromD3D10BufferKHR (
    cl_context context, cl_mem_flags flags,
    ID3D10Buffer *resource, cl_int *errcode_ret)
flags: CL_MEM_(READ, WRITE)_ONLY, CL_MEM_READ_WRITE
```

```
cl_mem clCreateFromD3D10Texture2DKHR (
    cl_context context, cl_mem_flags flags,
    ID3D10Texture2D *resource, UINT subresource,
    cl_int *errcode_ret)
flags: See clCreateFromD3D10BufferKHR
```

```
cl_mem clCreateFromD3D10Texture3DKHR (
    cl_context context, cl_mem_flags flags,
    ID3D10Texture3D *resource,
    UINT subresource,
    cl_int *errcode_ret)
flags: See clCreateFromD3D10BufferKHR
```

```
cl_int clEnqueueAcquireD3D10ObjectsKHR (
    cl_command_queue command_queue,
    cl_uint num_objects, const cl_mem *mem_objects,
    cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list,
    cl_event *event)
```

```
cl_int clEnqueueReleaseD3D10ObjectsKHR (
    cl_command_queue command_queue,
    cl_uint num_objects, const cl_mem *mem_objects,
    cl_uint num_events_in_wait_list,
    const cl_event *event_wait_list,
    cl_event *event)
```



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.