clCreateBuffer

Creates a buffer object.

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

Parameters

context

> A valid OpenCL context used to create the buffer object.

flags

> A bit-field that is used to specify allocation and usage information such as the memory arena that should be used to allocate the buffer object and how it will be used. The following table describes the possible values for *flags*:

cl_mem_flags	Description
CL_MEM_READ_WRITE	This flag specifies that the memory object will be read and written by a kernel. This is the default.
CL_MEM_WRITE_ONLY	This flags specifies that the memory object will be written but not read by a kernel. Reading from a buffer or image object created with CL_MEM_WRITE_ONLY inside a kernel is undefined.
CL_MEM_READ_ONLY	This flag specifies that the memory object is a read-only memory object when used inside a kernel. Writing to a buffer or image object created with CL_MEM_READ_ONLY inside a kernel is undefined.
CL_MEM_USE_HOST_PTR	This flag is valid only if *host_ptr* is not NULL. If specified, it indicates that the application wants the OpenCL implementation to use memory referenced by *host_ptr* as the storage bits for the memory object. OpenCL implementations are allowed to cache

cl_mem_flags	Description
	the buffer contents pointed to by *host_ptr* in device memory. This cached copy can be used when kernels are executed on a device. The result of OpenCL commands that operate on multiple buffer objects created with the same *host_ptr* or overlapping host regions is considered to be undefined.
CL_MEM_ALLOC_HOST_PTR	This flag specifies that the application wants the OpenCL implementation to allocate memory from host accessible memory. CL_MEM_ALLOC_HOST_PTR and CL_MEM_USE_HOST_PTR are mutually exclusive.
CL_MEM_COPY_HOST_PTR	This flag is valid only if *host_ptr* is not NULL. If specified, it indicates that the application wants the OpenCL implementation to allocate memory for the memory object and copy the data from memory referenced by *host_ptr*. CL_MEM_COPY_HOST_PTR and CL_MEM_USE_HOST_PTR are mutually exclusive. CL_MEM_COPY_HOST_PTR can be used with CL_MEM_ALLOC_HOST_PTR to initialize the contents of the cl_mem object allocated using host-accessible (e.g. PCIe) memory.

size

> The size in bytes of the buffer memory object to be allocated.

host_ptr

> A pointer to the buffer data that may already be allocated by the application. The size of the buffer that *host_ptr* points to must be greater than or equal to the *size* bytes.

errcode_ret

> Returns an appropriate error code. If *errcode_ret* is NULL, no error code is returned.

Errors

Returns a valid non-zero buffer object and *errcode_ret* is set to CL_SUCCESS if the buffer object is created successfully. Otherwise, it returns a NULL value with one of the following error values returned in *errcode_ret*:

- CL_INVALID_CONTEXT if *context* is not a valid context

- CL_INVALID_VALUE if values specified in *flags* are not valid.

- CL_INVALID_BUFFER_SIZE if *size* is 0 or is greater than CL_DEVICE_MAX_MEM_ALLOC_SIZE value specified in table of OpenCL Device Queries for clGetDeviceInfo for all devices in *context*.

- CL_INVALID_HOST_PTR if *host_ptr* is NULL and CL_MEM_USE_HOST_PTR or CL_MEM_COPY_HOST_PTR are set in *flags* or if *host_ptr* is not NULL but CL_MEM_COPY_HOST_PTR or CL_MEM_USE_HOST_PTR are not set in *flags*.

- CL_MEM_OBJECT_ALLOCATION_FAILURE if there is a failure to allocate memory for buffer object.

- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

Specification

📕 OpenCL Specification

Also see

clEnqueueReadBuffer, clEnqueueWriteBuffer, clEnqueueCopyBuffer

KHRONOS
GROUP