```
1. m2s_int m2sGetPlatformIDs (m2s_uint num_entries,
                              m2s_platform_id *platforms,
                              m2s_uint *num_platforms)
```

Parameters

num_entries

The number of supported m2s_platform_id entries. If platforms is not NULL, the num_entries must be greater than zero.

platforms

Returns a list of OpenCL platforms found. The m2s_platform_id values returned in platforms can be used to identify a specific OpenCL platform. If platforms argument is NULL, this argument is ignored.

num_platforms

Returns the number of the OpenCL platforms available. If num_platforms is NULL, this argument is ignored.

Return value

ERROR

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Description

m2sGetPlatformIDs can be used for obtaining the list of platforms supported by OpenCL or the number of platforms. If you want to obtain the number of platforms, then use this API with num_entries as 0 and platforms as NULL. In case of obtaining the list of platforms, num_platforms will be NULL.

2.
```
m2s_int m2sGetDeviceIDs (m2s_platform_id platform,
                         m2s_device_type device_type,
                         m2s_uint num_entries,
                         m2s_device_id *device,
                         m2s_uint *num_devices)
```

Parameters

platform

Refers to the platform ID returned by m2sGetPlatformIDs.

device_type

Indicates the type of OpenCL device. The valid device_types are specified at 1.1.

num_entries

The number of m2s_device_id entries that can be supported.

device

m2s device that contain a list of OpenCL devices found. If device is NULL, this argument is ignored.

num_devices

The number of OpenCL devices supported that match device_type. If num_entries is NULL, this argument is ignored.

Return value

ERROR

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Description

m2sGetDeviceIDs can be used for obtaining the m2s device or the number of devices supported by OpenCL. If you want to obtain the number of devices, then use this API with num_entries as 0 and devices as NULL. In case of obtaining the m2s devices, num_devices will be NULL.

3.
```
m2s_context m2sCreateContext (const m2s_context_properties *properties,
                               const m2s_device_id device,
                               void *pfn_notify,
                               void *user_data,
                               m2s_int *errcode_ret)
```

Parameters

properties

NULL

device

Indicates the m2s_device_id that can contain multiple cl_device_ids. You can get m2s_device_id, using API m2sGetDeviceIDs.

pfn_notify

NULL

user_data

NULL

errcode_ret

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Return value

m2s_context

Returns m2s_context for the OpenCL runtime managing objects, such as command queues, memory, program, and kernel objects.

Description

m2sCreateContext creates and return an m2s_context.

4. m2s_command_queue m2sCreateCommandQueue (m2s_context context,
                                            m2s_device_id device,
                                            m2s_command_queue_properties properties,
                                            m2s_int *errcode_ret)

Parameters

context

Must be a valid M2S context.

device

Indicates the m2s_device_id that can contain multiple cl_device_ids. You can get m2s_device_id, using API m2sGetDeviceIDs.

properties

NULL: command queue will be executed as in-order queue.

M2S_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE: command queue will be executed as out-of-order queue.

errcode_ret

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Return value

m2s_command_queue

Returns m2s_command_queue for queuing a set of operations.

Description

m2sCreateCommandQueue creates and return an m2s_command_queue with multiple cl_command_queues.

5.
```
m2s_program m2sCreateProgramWithSource (m2s_context context,
                                         m2s_uint count,
                                         const char **strings,
                                         const size_t *lengths,
                                         m2s_int *errcode_ret)
```

Parameters

context

Must be a valid M2S context.

count

Indicates the number of the source code.

strings

In each string, there is source code for kernel. The source code must be written as char and terminated with NULL.

lengths

An array with the length of the source code in each string.

errcode_ret

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Return value

m2s_program

Returns m2s_program for the OpenCL.

Description

m2sCreateProgramWithSource creates and return an m2s_program. However, still m2s_program is not executable.

6.
```
m2s_int m2sBuildProgram (m2s_program program,
                         const m2s_device_id *device,
                         const char *options,
                         void *pfn_notify,
                         void *user_data)
```

Parameters

program

The program object.

device

Indicates the m2s_device_id that can contain multiple cl_device_ids. You can get m2s_device_id, using API m2sGetDeviceIDs.

options

A pointer to a null-terminated string of chars that describes the build options to be used for building the program executable. If you want to see detail, refer to OpenCL reference page.

pfn_notify

NULL

user_data

NULL

Return value

ERROR

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Description

m2sBuildProgram build the input m2s_program for make kernel objects.

7. `m2s_kernel m2sCreateKernel (m2s_program program,`
```
                               const char *kernel_name,
                               m2s_int *errcode_ret)
```

Parameters

program

A program object with a successfully built executable.

kernel_name

A function name in the program declared with the __kernel qualifier.

errcode_ret

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Return value

m2s_kernel

Returns m2s_kernel which is executed on the OpenCL device.

Description

m2sCreateKernel creates and return an m2s_kernel. This kernel object will be executed on m2s_devices.

```
8. m2s_mem m2sCreateBuffer (m2s_context context,
                            m2s_device_id * hint,
                            m2s_mem_flags flags,
                            size_t size,
                            void *host_ptr,
                            m2s_int *errcode_ret)
```

Parameters

context

A valid m2s_context used to create the buffer object.

hint

A hint for the memory division.

flags

give information for memory usage such as read only, write only, read write, and so on.

M2S_MEM_READ_WRITE: device memory can be read and written.

M2S_MEM_READ_ONLY: device memory can be read only.

M2S_MEM_WRITE_ONLY: device memory can be written only.

M2S_MEM_COPY_HOST_PTR: device memory data copied using host memory pointer.

size

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

host_ptr

A pointer to the buffer that may already be allocated by the application.

errcode_ret

Returns CL_SUCCESS when the function is executed successfully. Otherwise return error code.

Return value

m2s_mem

Returns m2s_mem which is executed on the OpenCL device.

Description

m2sCreateBuffer creates and return an m2s_mem which contains multiple cl_mems. And m2s_mem keeps hint for memory division. So, when execute kernel, divide data as hint says.

9.
```
m2s_int m2sEnqueueWriteBuffer (m2s_command_queue command_queue,
                               m2s_mem buffer,
                               m2s_bool blocking_write,
                               size_t offset,
                               size_t size,
                               const void *ptr,
                               m2s_uint num_events_in_wait_list,
                               const m2s_event *event_wait_list,
                               m2s_event *event)
```

Parameters

command_queue

A valid m2s_command_queue for queuing.

buffer

Refers to a valid buffer object.

blocking_write

M2S_TRUE (CL_TRUE): block mode

M2S_FALSE(CL_FALSE): non-blocking mode

offset

The offset in bytes in the buffer object to write to.

size

The size in bytes of data being written.

ptr

The pointer to buffer in host memory where data exist to be written.

num_events_in_wait_list

The number of event list.

event_wait_list

A list of events.

event

Returns an event object.

Return value

ERROR

Returns CL_SUCCESS when executed successfully. Otherwise return error code.

Description

m2sEnqueueWriteBuffer enqueue write operation to a specific command queue.

10.
```
m2s_int m2sEnqueueReadBuffer (m2s_command_queue command_queue,
                              m2s_mem buffer,
                              m2s_bool blocking_write,
                              size_t offset,
                              size_t size,
                              const void *ptr,
                              m2s_uint num_events_in_wait_list,
                              const m2s_event *event_wait_list,
                              m2s_event *event)
```

Parameters

command_queue

A valid m2s_command_queue for queuing.

buffer

Refers to a valid buffer object.

blocking_write

M2S_TRUE (CL_TRUE): block mode

M2S_FALSE(CL_FALSE): non-blocking mode

offset

The offset in bytes in the buffer object to read from.

size

The size in bytes of data being read.

ptr

The pointer to buffer in host memory where data is to be read into.

num_events_in_wait_list

The number of event list.

event_wait_list

A list of events.

event

Returns an event object.

Return value

ERROR

Returns CL_SUCCESS when executed successfully. Otherwise return error code.

Description

m2sEnqueueWriteBuffer enqueue read operation to a specific command queue.

11. `m2s_int m2sSetKernelArg (m2s_kernel kernel,`
```
                             m2s_device_id *device,
                             m2s_uint arg_index,
                             size_t arg_size,
                             m2s_mem *arg_value)
```

Parameters

kernel

A valid kernel object to be executed.

device

Refers to a valid buffer object.

arg_index

M2S_TRUE (CL_TRUE): block mode

M2S_FALSE(CL_FALSE): non-blocking mode

arg_size

The offset in bytes in the buffer object to read from.

arg_value

The size in bytes of data being read.

Return value

ERROR

Returns CL_SUCCESS when executed successfully. Otherwise return error code.

Description

m2sEnqueueWriteBuffer enqueue read operation to a specific command queue.

10. m2s_int m2sEnqueueReadBuffer (m2s_command_queue command_queue,

```
                                  m2s_mem buffer,
                                  m2s_bool blocking_write,
                                  size_t offset,
                                  size_t size,
                                  const void *ptr,
                                  m2s_uint num_events_in_wait_list,
                                  const m2s_event *event_wait_list,
                                  m2s_event *event)
```

Parameters

command_queue

A valid m2s_command_queue for queuing.

buffer

Refers to a valid buffer object.

blocking_write

M2S_TRUE (CL_TRUE): block mode

M2S_FALSE(CL_FALSE): non-blocking mode

offset

The offset in bytes in the buffer object to read from.

size

The size in bytes of data being read.

ptr

The pointer to buffer in host memory where data is to be read into.

num_events_in_wait_list

The number of event list.

event_wait_list

A list of events.

event

Returns an event object.

Return value

ERROR

Returns CL_SUCCESS when executed successfully. Otherwise return error code.

Description

m2sEnqueueWriteBuffer enqueue read operation to a specific command queue