|  |  |  |
| --- | --- | --- |
|  | **STDCL 1.4 Fortran Quick Reference Card** | |
| STDCL provides a simplified interface to OpenCL designed in a style familiar to conventional UNIX/C programmers. | The STDCL interface provides support for default contexts, a dynamic CL program loader, memory management, kernel management, and asynchronous operations. |

|  |  |  |
| --- | --- | --- |
| **Default CL Contexts**  type(C\_PTR) **~ CLCONTEXT\* stddev**  type(C\_PTR) **~ CLCONTEXT\* stdcpu**  type(C\_PTR) **~ CLCONTEXT\* stdgpu**  type(C\_PTR) **~ CLCONTEXT\* stdrpu**  Default context for **[** all **|** CPU **|** GPU **|** RPU **]** OpenCL supported devices.  **Platform**  **integer(C\_INT)**  **function clgetndev( *clcontext* )**  **type(C\_PTR) *clcontext***  Returns number of devices in context.  **Dynamic CL Program Loader**  **type(C\_PTR)**  **function clopen( *clcontext, filename, flags* )**  **type(C\_PTR) *clcontext***  **character(kind=C\_CHAR) *filename***  **integer(C\_INT) *flags***  ***flags*: CLLD\_NOW, CLLD\_NOBUILD**  Build the OpenCL device program and return a handle to the program.  **type(C\_PTR)**  **function clsopen( *clcontext, srcstr, flags* )**  **type(C\_PTR) *clcontext***  **character(kind=C\_CHAR) *srcstr***  **integer(C\_INT) *flags***  ***flags*: CLLD\_NOW, CLLD\_NOBUILD**  Build the OpenCL device program and return a handle to the program.  **type(C\_PTR) ~ cl\_kernel**  **function clsym( *clcontext, handle, symbol, flags* )**  **type(C\_PTR) clcontext**  **type(C\_PTR) handle**  **character(kind=C\_CHAR) symbol**  **integer(C\_INT) flags**  ***flags*: CLLD\_NOW**  Returns the kernel object identified by name from the compiled OpenCL device program.  **integer(C\_INT)**  **function clclose( *clcontext, handle* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *handle***  Close the OpenCL device program and release associated resources.  **type(C\_PTR)**  **function clbuild( *clcontext, handle, options, flag* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *handle***  **character(kind=C\_CHAR) *options***  **integer(C\_INT) *flags***  Build the OpenCL device program and return the handle to the program. | **Memory Management**  **type(C\_PTR)**  **function clmalloc( *clcontext, size, flags* )**  **type(C\_PTR) *clcontext***  **integer(C\_SIZE\_T) *size***  **integer(C\_INT) *flag***  ***flags:* CL\_MEM\_DETACHED**  Allocate memory that can be shared across OpenCL devices.  **type(C\_PTR)**  **function clmrealloc( *clcontext, ptr, size, flags* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *ptr***  **integer(C\_SIZE\_T) *size***  **integer(C\_INT) *flags***  ***flags:* CL\_MEM\_DETACHED**  Re-allocate (re-size) memory that can be shared across OpenCL devices.  **integer(C\_INT)**  **function clfree( *ptr* )**  **type(C\_PTR) *ptr***  Free device-shareable memory allocated with clmalloc() or an equivalent call.  **type(C\_PTR) ~ cl\_event**  **function clmsync( *clcontext, devnum, ptr, flags* )**  **type(C\_PTR) *clcontext***  **integer(C\_INT) devnum**  **type(C\_PTR) *ptr***  **integer(C\_INT) *flags***  ***flags:* CL\_MEM\_HOST | CL\_MEM\_DEVICE,**  **CL\_EVENT\_WAIT | CL\_EVENT\_NOWAIT,**  **CL\_EVENT\_NORELEASE**  Synchronize memory on host or OpenCL device, performing a memory copy as necessary.  **type(C\_PTR) ~ cl\_event**  **function clmcopy(*clcontext, devnum, src, dst, flags*)**  **type(C\_PTR) *clcontext***  **integer(C\_INT) devnum**  **Type(C\_PTR) *src***  **Type(C\_PTR) *dst***  **integer(C\_INT) *flags***  ***flags:* CL\_EVENT\_WAIT | CL\_EVENT\_NOWAIT,**  **CL\_EVENT\_NORELEASE**  Copy memory on an OpenCL device.  **integer(C\_INT)**  **function clmattach( *clcontext, ptr* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *ptr***  Attach device-shareable memory to context.  **integer(C\_INT)**  **function clmdetach( *ptr* )**  **type(C\_PTR) *ptr***  Detach device-shareable memory from context. | **integer(C\_SIZE\_T)**  **function clsizeofmem( *ptr* )**  **type(C\_PTR) *ptr***  Return the size of device-shareable memory allocated with clmalloc() or an equivalent call.  **Kernel Management**  **type(clndrange\_struct)**  **function clndrange\_init[1|2|3] d( *gtoff0, gtsz0, ltsz0* [ *,gtoff1, gtsz1, ltsz1,* [ *,gtoff2, gtsz2, ltsz2* ] ])**  **integer(C\_INT) *gtoff0* [ *,gtoff1* [ *,gtoff2* ] ]**  **integer(C\_INT) *gtsz0* [ *,gtsz1* [ *,gtsz2* ] ]**  **integer(C\_INT) *ltsz0* [ *,ltsz1* [ *,ltsz2* ] ]**  Initialize N-dimensional range.  **integer(C\_INT)**  **function clarg\_set( *clcontext, krn, argnum, arg* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *krn***  **integer(C\_INT) argnum**  **Tn *arg***  Set intrinsic argument of kernel.  **integer(C\_INT)**  **function clarg\_set\_global( *clcontext, krn, argnum, ptr* )**  **type(C\_PTR) *clcontext***  **type(C\_PTR) *krn***  **integer(C\_INT) argnum**  **type(C\_PTR) *ptr***  Set pointer argument of kernel.  **type(C\_PTR) ~ cl\_event**  **function clfork( *clcontext, devnum, krn, ndr\_ptr, flags* )**  **type(C\_PTR) *clcontext***  **integer(C\_INT) devnum**  **type(C\_PTR) *krn***  **type(C\_PTR) *ndr\_ptr* ~ C\_LOC(clndrange\_struct)**  **integer(C\_INT) *flags***  ***flags:* CL\_EVENT\_WAIT | CL\_EVENT\_NOWAIT,**  **CL\_EVENT\_NORELEASE**  Fork kernel for execution on OpenCL device.  **Synchronization**  **type(C\_PTR)**  **function clflush( *clcontext, devnum, flags* )**  **type(C\_PTR) *clcontext***  **integer(C\_INT) devnum**  **integer(C\_INT) *flags***  ***flags:* CL\_KERNEL\_EVENT, CL\_MEM\_EVENT**  **CL\_ALL\_EVENT, CL\_EVENT\_NORELEASE**  Flush all enqueued operations (non-blocking).  **type(C\_PTR)**  **function clwait( *clcontext, devnum, flags* )**  **type(C\_PTR) *clcontext***  **integer(C\_INT) devnum**  **integer(C\_INT) *flags***  ***flags:* CL\_KERNEL\_EVENT, CL\_MEM\_EVENT**  **CL\_ALL\_EVENT, CL\_EVENT\_NORELEASE**  Block on all enqueued operations. |

**Notation:**

**~ type** indicates the opaque type for which the C\_PTR is used as a proxy since Fortran does not support type aliasing.

**[**a **|** b **| … ]**  indicates a choice between several alternatives and is not part of the syntax.