HIPxx

0.0.1

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

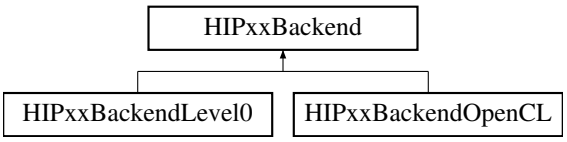
Class Documentation

4.1 HIPxxBackend Class Reference

Primary object to interact with the backend.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxBackend:



Public Member Functions

- HIPxxBackend ()

  Construct a new HIPxxBackend object.
- ~HIPxxBackend ()

  Destroy the HIPxxBackend objectk.
- virtual void initialize (std::string platform_str, std::string device_type_str, std::string device_ids_str)
- virtual void **initialize** ()=0
- virtual void **uninitialize** ()=0
- std::vector< HIPxxQueue * > & getQueues ()

  Get the Queues object.
- HIPxxQueue * getActiveQueue ()

  Get the Active Queue object.
- HIPxxContext * getActiveContext ()

  Get the Active Context object. Returns the context of the active queue.
- HIPxxDevice * getActiveDevice ()

  Get the Active Device object. Returns the device of the active queue.
- void setActiveDevice (HIPxxDevice *hipxx_dev)

  Set the active device. Sets the active queue to this device's first/default/primary queue.
- std::vector< HIPxxDevice * > & **getDevices** ()
- size_t getNumDevices ()

Get the Num Devices object.

- std::vector< std::string * > & getModulesStr ()

  Get the vector of registered modules (in string/binary format)

- void addContext (HIPxxContext *ctx_in)

  Add a context to this backend.

- void addQueue (HIPxxQueue *q_in)

  Add a context to this backend.

- void addDevice (HIPxxDevice *dev_in)

  Add a device to this backend.

- void registerModuleStr (std::string *mod_str)
- void unregisterModuleStr (std::string *mod_str)
- hipError_t configureCall (dim3 grid, dim3 block, size_t shared, hipStream_t q)

  Configure an upcoming kernel call.

- hipError_t setArg (const void *arg, size_t size, size_t offset)

  Set the Arg object.

- virtual bool registerFunctionAsKernel (std::string *module_str, const void *host_f_ptr, const char *host_f_↩ name)

  Register this function as a kernel for all devices initialized in this backend.

- HIPxxDevice * findDeviceMatchingProps (const hipDeviceProp_t *props)

  Return a device which meets or exceeds the requirements.

- hipError_t addModule (HIPxxModule *hipxx_module)

  Add a HIPxxModule to every initialized device.

- hipError_t removeModule (HIPxxModule *hipxx_module)

  Remove this module from every device.

Public Attributes

- std::stack< HIPxxExecItem * > **hipxx_execstack**
- std::vector< HIPxxContext * > **hipxx_contexts**
- std::vector< HIPxxQueue * > **hipxx_queues**
- std::vector< HIPxxDevice * > **hipxx_devices**

Static Public Attributes

- static thread_local hipError_t **tls_last_error** = hipSuccess
- static thread_local HIPxxContext * **tls_active_ctx**

Protected Attributes

- std::vector< std::string * > modules_str

  hipxx_modules stored in binary representation. During compilation each translation unit is parsed for functions that are marked for execution on the device. These functions are then compiled to device code and stored in binary representation.

- std::mutex **mtx**
- HIPxxContext * **active_ctx**
- HIPxxDevice * **active_dev**
- HIPxxQueue * **active_q**

4.1.1 Detailed Description

Primary object to interact with the backend.

4.1.2 Member Function Documentation

4.1.2.1 addContext()

```
void HIPxxBackend::addContext (
            HIPxxContext * ctx_in )
```

Add a context to this backend.

Parameters

ctx_in	

4.1.2.2 addDevice()

```
void HIPxxBackend::addDevice (
            HIPxxDevice * dev_in )
```

Add a device to this backend.

Parameters

dev_in	

4.1.2.3 addModule()

```
hipError_t HIPxxBackend::addModule (
            HIPxxModule * hipxx_module )
```

Add a HIPxxModule to every initialized device.

Parameters

hipxx_module	pointer to HIPxxModule object

Returns

hipError_t

4.1.2.4 addQueue()

```
void HIPxxBackend::addQueue (
            HIPxxQueue * q_in )
```

Add a context to this backend.

Parameters

q_in	

4.1.2.5 configureCall()

```
hipError_t HIPxxBackend::configureCall (
            dim3 grid,
            dim3 block,
            size_t shared,
            hipStream_t q )
```

Configure an upcoming kernel call.

Parameters

grid	
block	
shared	
q	

Returns

hipError_t

4.1.2.6 findDeviceMatchingProps()

```
HIPxxDevice * HIPxxBackend::findDeviceMatchingProps (
            const hipDeviceProp_t * props )
```

Return a device which meets or exceeds the requirements.

Parameters

props	

Returns

HIPxxDevice*

4.1.2.7 getActiveContext()

```
HIPxxContext * HIPxxBackend::getActiveContext ( )
```

Get the Active Context object. Returns the context of the active queue.

Returns

HIPxxContext*

4.1.2.8 getActiveDevice()

```
HIPxxDevice * HIPxxBackend::getActiveDevice ( )
```

Get the Active Device object. Returns the device of the active queue.

Returns

HIPxxDevice*

4.1.2.9 getActiveQueue()

```
HIPxxQueue * HIPxxBackend::getActiveQueue ( )
```

Get the Active Queue object.

Returns

HIPxxQueue*

4.1.2.10 getModulesStr()

```
std::vector< std::string * > & HIPxxBackend::getModulesStr ( )
```

Get the vector of registered modules (in string/binary format)

Returns

std::vector<std::string*>&

4.1.2.11 getNumDevices()

```
size_t HIPxxBackend::getNumDevices ( )
```

Get the Num Devices object.

Returns

size_t

4.1.2.12 getQueues()

```
std::vector< HIPxxQueue * > & HIPxxBackend::getQueues ( )
```

Get the Queues object.

Returns

std::vector<HIPxxQueue*>&

4.1.2.13 initialize()

```
void HIPxxBackend::initialize (
            std::string platform_str,
            std::string device_type_str,
            std::string device_ids_str )  [virtual]
```

Parameters

platform_str	
device_type_str	
device_ids_str	

Reimplemented in HIPxxBackendOpenCL, and HIPxxBackendLevel0.

4.1.2.14 registerFunctionAsKernel()

```
bool HIPxxBackend::registerFunctionAsKernel (
            std::string * module_str,
            const void * host_f_ptr,
            const char * host_f_name )  [virtual]
```

Register this function as a kernel for all devices initialized in this backend.

Parameters

module_str	
host_f_ptr	
host_f_name	

Returns

true

false

Parameters

module_str	
HostFunctionPtr	
FunctionName	

Returns

true

false

4.1.2.15 registerModuleStr()

```
void HIPxxBackend::registerModuleStr (
            std::string * mod_str )
```

Parameters

mod_str	

4.1.2.16 removeModule()

```
hipError_t HIPxxBackend::removeModule (
            HIPxxModule * hipxx_module )
```

Remove this module from every device.

Parameters

hipxx_module	pointer to the module which is to be removed

Returns

hipError_t

4.1.2.17 setActiveDevice()

```
void HIPxxBackend::setActiveDevice (
            HIPxxDevice * hipxx_dev )
```

Set the active device. Sets the active queue to this device's first/default/primary queue.

Parameters

hipxx_dev	

4.1.2.18 setArg()

```
hipError_t HIPxxBackend::setArg (
            const void * arg,
            size_t size,
            size_t offset )
```

Set the Arg object.

Parameters

arg	
size	
offset	

Returns

hipError_t

4.1.2.19 unregisterModuleStr()

```
void HIPxxBackend::unregisterModuleStr (
            std::string * mod_str )
```

Parameters

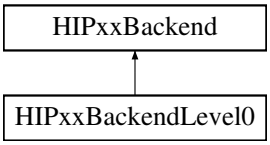
mod_str	

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.2 HIPxxBackendLevel0 Class Reference

Inheritance diagram for HIPxxBackendLevel0:



Public Member Functions

- virtual void initialize (std::string HIPxxPlatformStr, std::string HIPxxDeviceTypeStr, std::string HIPxxDeviceStr) override
- virtual void **initialize** () override
- void **uninitialize** () override

Additional Inherited Members

4.2.1 Member Function Documentation

4.2.1.1 initialize()

```
virtual void HIPxxBackendLevel0::initialize (
            std::string platform_str,
            std::string device_type_str,
            std::string device_ids_str )  [inline], [override], [virtual]
```

Parameters

platform_str	
device_type_str	
device_ids_str	

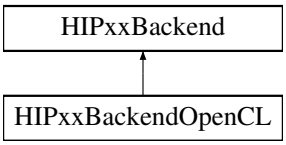
Reimplemented from HIPxxBackend.

The documentation for this class was generated from the following file:

- src/backend/Level0/Level0Backend.hh

4.3 HIPxxBackendOpenCL Class Reference

Inheritance diagram for HIPxxBackendOpenCL:



Public Member Functions

- void initialize (std::string HIPxxPlatformStr, std::string HIPxxDeviceTypeStr, std::string HIPxxDeviceStr) override
- virtual void **initialize** () override
- void **uninitialize** () override

Additional Inherited Members

4.3.1 Member Function Documentation

4.3.1.1 initialize()

```
void HIPxxBackendOpenCL::initialize (
            std::string platform_str,
            std::string device_type_str,
            std::string device_ids_str )  [inline], [override], [virtual]
```

Parameters

platform_str	
device_type_str	
device_ids_str	

Reimplemented from HIPxxBackend.

The documentation for this class was generated from the following file:

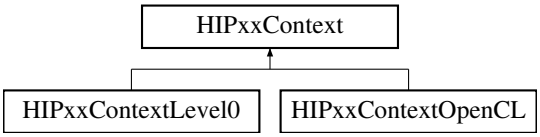
- src/backend/OpenCL/HIPxxBackendOpenCL.hh

4.4 HIPxxContext Class Reference

Context class Contexts contain execution queues and are created on top of a single or multiple devices. Provides for creation of additional queues, events, and interaction with devices.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxContext:



Public Member Functions

- bool addDevice (HIPxxDevice ∗dev)

  Add a device to this context.
- void **addQueue** (HIPxxQueue ∗q)
- HIPxxQueue ∗ **getDefaultQueue** ()
- hipStream_t **findQueue** (hipStream_t stream)
- std::vector< HIPxxDevice ∗ > & **getDevices** ()
- std::vector< HIPxxQueue ∗ > & **getQueues** ()
- virtual void ∗ **allocate** (size_t size)=0
- virtual void ∗ **allocate** (size_t size, HIPxxMemoryType mem_type)
- virtual void ∗ **allocate** (size_t size, size_t alignment, HIPxxMemoryType mem_type)
- bool **free** (void ∗ptr)
- virtual hipError_t **memCopy** (void ∗dst, const void ∗src, size_t size, hipStream_t stream)
- hipError_t **launchHostFunc** (const void ∗HostFunction)
- void **finishAll** ()
- bool **findPointerInfo** (hipDeviceptr_t ∗pbase, size_t ∗psize, hipDeviceptr_t dptr)
- unsigned int **getFlags** ()
- bool **setFlags** (unsigned int flags)
- void **reset** ()
- HIPxxContext ∗ **retain** ()
- bool **recordEvent** (HIPxxQueue ∗q, HIPxxEvent ∗event)
- size_t **getPointerSize** (void ∗ptr)
- HIPxxTexture ∗ **createImage** (hipResourceDesc ∗resDesc, hipTextureDesc ∗texDesc)

Protected Attributes

- std::vector< HIPxxDevice ∗ > **hipxx_devices**
- std::vector< HIPxxQueue ∗ > **hipxx_queues**
- std::mutex **mtx**

4.4.1 Detailed Description

Context class Contexts contain execution queues and are created on top of a single or multiple devices. Provides for creation of additional queues, events, and interaction with devices.

4.4.2 Member Function Documentation

4.4.2.1 addDevice()

```
bool HIPxxContext::addDevice (
            HIPxxDevice * dev )
```

Add a device to this context.

Parameters

dev	pointer to HIPxxDevice object

Returns

true if device was added successfully

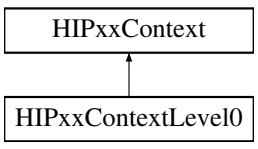
false upon failure

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.5 HIPxxContextLevel0 Class Reference

Inheritance diagram for HIPxxContextLevel0:



Public Member Functions

- ze_command_list_handle_t **get_cmd_list** ()
- **HIPxxContextLevel0** (ze_context_handle_t &&_ze_ctx)
- void * **allocate** (size_t size, size_t alignment, LZMemoryType memTy)
- virtual void * **allocate** (size_t size) override
- ze_context_handle_t & **get** ()
- virtual hipError_t **memCopy** (void *dst, const void *src, size_t size, hipStream_t stream) override

Public Attributes

- ze_command_list_handle_t **ze_cmd_list**

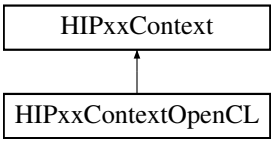
Additional Inherited Members

The documentation for this class was generated from the following files:

- src/backend/Level0/Level0Backend.hh
- src/backend/Level0/Level0Backend.cc

4.6 HIPxxContextOpenCL Class Reference

Inheritance diagram for HIPxxContextOpenCL:



Public Member Functions

- **HIPxxContextOpenCL** (cl::Context ∗ctx_in)
- void ∗ **allocate** (size_t size) override
- virtual hipError_t **memCopy** (void ∗dst, const void ∗src, size_t size, hipStream_t stream) override
- cl::Context ∗ **get** ()

Public Attributes

- SVMemoryRegion **svm_memory**
- cl::Context ∗ **cl_ctx**

Additional Inherited Members

The documentation for this class was generated from the following files:

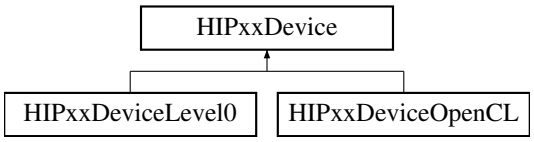
- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/HIPxxBackendOpenCL.cc

4.7 HIPxxDevice Class Reference

Compute device class.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxDevice:



Public Member Functions

- HIPxxDevice ()

  Construct a new HIPxxDevice object.
- ~HIPxxDevice ()

  Destroy the HIPxxDevice object.
- std::vector< HIPxxKernel * > & getKernels ()

  Get the Kernels object.
- virtual void populateDeviceProperties ()=0

  Use a backend to populate device properties such as memory available, frequencies, etc.
- void copyDeviceProperties (hipDeviceProp_t *prop)

  Query the device for properties.
- HIPxxKernel * findKernelByHostPtr (const void *hostPtr)

  Use the host function pointer to retrieve the kernel.
- HIPxxContext * getContext ()

  Get the context object.
- void addQueue (HIPxxQueue *hipxx_queue_)

  Construct an additional queue for this device.
- std::vector< HIPxxQueue * > getQueues ()

  Get the Queues object.
- HIPxxQueue * getActiveQueue ()

  HIP API allows for setting the active device, not the active queue so active device's active queue is always it's 0th/default/primary queue.
- bool removeQueue (HIPxxQueue *q)

  Remove a queue from this device's queue vector.
- int getDeviceId ()

  Get the integer ID of this device as it appears in the Backend's hipxx_devices list.
- virtual std::string getName ()=0

  Get the device name.
- bool **allocate** (size_t bytes)
- bool **free** (size_t bytes)
- virtual void reset ()=0

  Reset the device.
- int **getAttr** (int *pi, hipDeviceAttribute_t attr)
- size_t getGlobalMemSize ()

  Get the total global memory available for this device.
- void **setCacheConfig** (hipFuncCache_t)

- hipFuncCache_t **getCacheConfig** ()
- void **setSharedMemConfig** (hipSharedMemConfig config)
- hipSharedMemConfig **getSharedMemConfig** ()
- void **setFuncCacheConfig** (const void *func, hipFuncCache_t config)
- bool hasPCIBusId (int pciDomainID, int pciBusID, int pciDeviceID)

  Check if the current device has same PCI bus ID as the one given by input.
- int getPeerAccess (HIPxxDevice *peerDevice)

  Get peer-accesability between this and another device.
- hipError_t setPeerAccess (HIPxxDevice *peer, int flags, bool canAccessPeer)

  Set access between this and another device.
- size_t getUsedGlobalMem ()

  Get the total used global memory.
- HIPxxDeviceVar * **getDynGlobalVar** (const void *host_var_ptr)
- HIPxxDeviceVar * **getStatGlobalVar** (const void *host_var_ptr)
- HIPxxDeviceVar * **getGlobalVar** (const void *host_var_ptr)
- void registerFunctionAsKernel (std::string *module_str, const void *host_f_ptr, const char *host_f_name)

  Take the module source, compile the kernels and associate the host function pointer with a kernel whose name matches host function name.

Public Attributes

- std::vector< std::string * > modules_str

  hipxx_modules in binary representation
- std::vector< HIPxxModule * > hipxx_modules

  hipxx_modules in parsed representation
- std::unordered_map< const void *, std::string * > host_f_ptr_to_module_str_map

  Map host pointer to module in binary representation.
- std::unordered_map< const void *, HIPxxModule * > host_f_ptr_to_hipxxmodule_map

  Map host pointer to module in parsed representation.
- std::unordered_map< const void *, std::string > host_f_ptr_to_host_f_name_map

  Map host pointer to a function name.
- std::unordered_map< const void *, HIPxxKernel * > host_ptr_to_hipxxkernel_map

  Map host pointer to HIPxxKernel.
- std::unordered_map< const void *, HIPxxDeviceVar * > host_var_ptr_to_hipxxdevicevar_stat
- std::unordered_map< const void *, HIPxxDeviceVar * > host_var_ptr_to_hipxxdevicevar_dyn
- int **idx**
- hipDeviceProp_t **hip_device_props**
- size_t **total_used_mem**
- size_t **max_used_mem**

Protected Attributes

- std::string **device_name**
- std::mutex **mtx**
- std::vector< HIPxxKernel * > **hipxx_kernels**
- HIPxxContext * **ctx**
- std::vector< HIPxxQueue * > **hipxx_queues**
- int **active_queue_id** = 0
- hipDeviceAttribute_t **attrs**

4.7.1 Detailed Description

Compute device class.

4.7.2 Member Function Documentation

4.7.2.1 addQueue()

```
void HIPxxDevice::addQueue (
            HIPxxQueue * hipxx_queue_ )
```

Construct an additional queue for this device.

Parameters

flags	
priority	

Returns

HIPxxQueue* pointer to the newly created queue (can also be found in hipxx_queues vector)

4.7.2.2 copyDeviceProperties()

```
void HIPxxDevice::copyDeviceProperties (
            hipDeviceProp_t * prop )
```

Query the device for properties.

Parameters

prop	

4.7.2.3 findKernelByHostPtr()

```
HIPxxKernel * HIPxxDevice::findKernelByHostPtr (
            const void * hostPtr )
```

Use the host function pointer to retrieve the kernel.

Parameters

hostPtr	

Returns

HIPxxKernel* HIPxxKernel associated with this host pointer

4.7.2.4 getActiveQueue()

```
HIPxxQueue* HIPxxDevice::getActiveQueue ( )
```

HIP API allows for setting the active device, not the active queue so active device's active queue is always it's 0th/default/primary queue.

Returns

HIPxxQueue*

4.7.2.5 getContext()

```
HIPxxContext * HIPxxDevice::getContext ( )
```

Get the context object.

Returns

HIPxxContext* pointer to the HIPxxContext object this HIPxxDevice was created with

4.7.2.6 getDeviceId()

```
int HIPxxDevice::getDeviceId ( )
```

Get the integer ID of this device as it appears in the Backend's hipxx_devices list.

Returns

int

4.7.2.7 getGlobalMemSize()

```
size_t HIPxxDevice::getGlobalMemSize ( )
```

Get the total global memory available for this device.

Returns

size_t

4.7.2.8 getKernels()

```
std::vector< HIPxxKernel * > & HIPxxDevice::getKernels ( )
```

Get the Kernels object.

Returns

std::vector<HIPxxKernel*>&

4.7.2.9 getName()

```
virtual std::string HIPxxDevice::getName ( )  [pure virtual]
```

Get the device name.

Returns

std::string

Implemented in HIPxxDeviceOpenCL, and HIPxxDeviceLevel0.

4.7.2.10 getPeerAccess()

```
int HIPxxDevice::getPeerAccess (
            HIPxxDevice * peerDevice )
```

Get peer-accesability between this and another device.

Parameters

peerDevice	

Returns

int

4.7.2.11 getQueues()

```
std::vector<HIPxxQueue*> HIPxxDevice::getQueues ( )
```

Get the Queues object.

Returns

std::vector<HIPxxQueue*>

4.7.2.12 getUsedGlobalMem()

```
size_t HIPxxDevice::getUsedGlobalMem ( )
```

Get the total used global memory.

Returns

size_t

4.7.2.13 hasPCIBusId()

```
bool HIPxxDevice::hasPCIBusId (
            int pciDomainID,
            int pciBusID,
            int pciDeviceID )
```

Check if the current device has same PCI bus ID as the one given by input.

Parameters

pciDomainID	
pciBusID	
pciDeviceID	

Returns

true

false

4.7.2.14 registerFunctionAsKernel()

```
void HIPxxDevice::registerFunctionAsKernel (
            std::string * module_str,
            const void * host_f_ptr,
            const char * host_f_name )
```

Take the module source, compile the kernels and associate the host function pointer with a kernel whose name matches host function name.

Parameters

module_str	Binary representation of the SPIR-V module
host_f_ptr	host function pointer
host_f_name	host function name

4.7.2.15 removeQueue()

```
bool HIPxxDevice::removeQueue (
            HIPxxQueue * q )
```

Remove a queue from this device's queue vector.

Parameters

q	

Returns

true

false

4.7.2.16 setPeerAccess()

```
hipError_t HIPxxDevice::setPeerAccess (
            HIPxxDevice * peer,
            int flags,
            bool canAccessPeer )
```

Set access between this and another device.

Parameters

peer	
flags	
canAccessPeer	

Returns

hipError_t

4.7.3 Member Data Documentation

4.7.3.1 host_var_ptr_to_hipxxdevicevar_dyn

```
std::unordered_map<const void*, HIPxxDeviceVar*> HIPxxDevice::host_var_ptr_to_hipxxdevicevar↩
_dyn
```

Map host variable address to device pointer and size for dynamically loaded global vars

4.7.3.2 host_var_ptr_to_hipxxdevicevar_stat

```
std::unordered_map<const void*, HIPxxDeviceVar*> HIPxxDevice::host_var_ptr_to_hipxxdevicevar↩
_stat
```

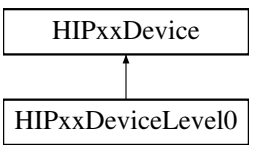
Map host variable address to device pointer and size for statically loaded global vars

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.8 HIPxxDeviceLevel0 Class Reference

Inheritance diagram for HIPxxDeviceLevel0:



Public Member Functions

- **HIPxxDeviceLevel0** (ze_device_handle_t &&ze_dev_, ze_context_handle_t ze_ctx_)
- virtual void populateDeviceProperties () override

  Use a backend to populate device properties such as memory available, frequencies, etc.
- virtual std::string getName () override

  Get the device name.
- ze_device_handle_t & **get** ()
- virtual void reset () override

  Reset the device.

Additional Inherited Members

4.8.1 Member Function Documentation

4.8.1.1 getName()

```
virtual std::string HIPxxDeviceLevel0::getName ( )  [inline], [override], [virtual]
```

Get the device name.

Returns

std::string

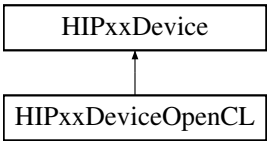
Implements HIPxxDevice.

The documentation for this class was generated from the following files:

- src/backend/Level0/Level0Backend.hh
- src/backend/Level0/Level0Backend.cc

4.9 HIPxxDeviceOpenCL Class Reference

Inheritance diagram for HIPxxDeviceOpenCL:



Public Member Functions

- **HIPxxDeviceOpenCL** (HIPxxContextOpenCL *hipxx_ctx, cl::Device *dev_in, int idx)
- cl::Device * **get** ()
- virtual void populateDeviceProperties () override

  Use a backend to populate device properties such as memory available, frequencies, etc.
- virtual std::string getName () override

  Get the device name.
- virtual void reset () override

  Reset the device.

Public Attributes

- cl::Device * **cl_dev**
- cl::Context * **cl_ctx**

Additional Inherited Members

4.9.1 Member Function Documentation

4.9.1.1 getName()

```
std::string HIPxxDeviceOpenCL::getName ( )  [override], [virtual]
```

Get the device name.

Returns

std::string

Implements HIPxxDevice.

The documentation for this class was generated from the following files:

- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/HIPxxBackendOpenCL.cc

4.10 HIPxxDeviceVar Class Reference

Public Member Functions

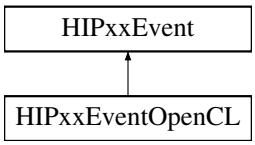
- **HIPxxDeviceVar** (std::string host_var_name_, void *dev_ptr_, size_t size)
- void * **getDevAddr** ()
- std::string **getName** ()
- size_t **getSize** ()

The documentation for this class was generated from the following file:

- src/HIPxxBackend.hh

4.11 HIPxxEvent Class Reference

Inheritance diagram for HIPxxEvent:



Public Member Functions

- HIPxxEvent (HIPxxContext *ctx_, HIPxxEventType flags_=HIPxxEventType::Default)

  HIPxxEvent constructor. Must always be created with some context.
- ~HIPxxEvent ()

  Destroy the HIPxxEvent object.
- virtual bool recordStream (HIPxxQueue *hipxx_queue_)

  Enqueue this event in a given HIPxxQueue.
- virtual bool wait ()

  Wait for this event to complete.
- virtual bool isFinished ()

  Query the event to see if it completed.
- virtual float getElapsedTime (HIPxxEvent *other)

  Calculate absolute difference between completion timestamps of this event and other.

Protected Member Functions

- HIPxxEvent ()=default

  hidden default constructor for HIPxxEvent. Only derived class constructor should be called.

Protected Attributes

- std::mutex **mutex**
- event_status_e **status**
- HIPxxEventType flags

  event bahavior modifier - valid values are hipEventDefault, hipEventBlockingSync, hipEventDisableTiming, hipEventInterprocess
- HIPxxContext * hipxx_context

  Events are always created with a context.

4.11.1 Member Function Documentation

4.11.1.1 getElapsedTime()

```
float HIPxxEvent::getElapsedTime (
            HIPxxEvent * other )  [virtual]
```

Calculate absolute difference between completion timestamps of this event and other.

Parameters

other	

Returns

float

4.11.1.2 isFinished()

```
bool HIPxxEvent::isFinished ( )  [virtual]
```

Query the event to see if it completed.

Returns

true

false

4.11.1.3 recordStream()

```
bool HIPxxEvent::recordStream (
            HIPxxQueue * hipxx_queue_ )  [virtual]
```

Enqueue this event in a given HIPxxQueue.

Parameters

*hipxx_queue_*	HIPxxQueue in which to enque this event

Returns

true

false

4.11.1.4 wait()

```
bool HIPxxEvent::wait ( )  [virtual]
```

Wait for this event to complete.

Returns

true

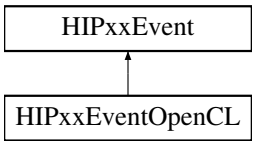
false

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.12 HIPxxEventOpenCL Class Reference

Inheritance diagram for HIPxxEventOpenCL:



Protected Attributes

- cl::Event * **cl_event**

Additional Inherited Members

The documentation for this class was generated from the following file:

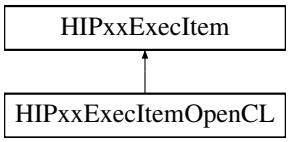
- src/backend/OpenCL/HIPxxBackendOpenCL.hh

4.13 HIPxxExecItem Class Reference

Contains kernel arguments and a queue on which to execute. Prior to kernel launch, the arguments are setup via HIPxxBackend::configureCall(). Because of this, we get the kernel last so the kernel so the launch() takes a kernel argument as opposed to queue receiving a HIPxxExecItem containing the kernel and arguments.

```
#include <HIPxxBackend.hh>
```

Inheritance diagram for HIPxxExecItem:



Public Member Functions

- HIPxxExecItem ()=delete

  Deleted default constructor Doesn't make sense for HIPxxExecItem to exist without arguments.
- HIPxxExecItem (dim3 grid_dim_, dim3 block_dim_, size_t shared_mem_, hipStream_t hipxx_queue_)

  Construct a new HIPxxExecItem object.
- HIPxxKernel * getKernel ()

  Get the Kernel object.
- HIPxxQueue * getQueue ()

  Get the Queue object.
- dim3 getGrid ()

  Get the Grid object.
- dim3 getBlock ()

  Get the Block object.
- void setArg (const void *arg, size_t size, size_t offset)

  Setup a single argument. gets called by hipSetupArgument calls to which are emitted by hip-clang.
- virtual hipError_t launch (HIPxxKernel *Kernel)

  Submit a kernel to the associated queue for execution. hipxx_queue must be set prior to this call.
- hipError_t launchByHostPtr (const void *hostPtr)

  Launch a kernel associated with a host function pointer. Looks up the HIPxxKernel associated with this pointer and calls launch()

Protected Attributes

- size_t **shared_mem**
- std::vector< uint8_t > **arg_data**
- std::vector< std::tuple< size_t, size_t > > **offset_sizes**
- dim3 **grid_dim**
- dim3 **block_dim**
- HIPxxQueue * **stream**
- HIPxxKernel * **hipxx_kernel**
- HIPxxQueue * **hipxx_queue**

4.13.1 Detailed Description

Contains kernel arguments and a queue on which to execute. Prior to kernel launch, the arguments are setup via HIPxxBackend::configureCall(). Because of this, we get the kernel last so the kernel so the launch() takes a kernel argument as opposed to queue receiving a HIPxxExecItem containing the kernel and arguments.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 HIPxxExecItem()

```
HIPxxExecItem::HIPxxExecItem (
            dim3 grid_dim_,
            dim3 block_dim_,
            size_t shared_mem_,
            hipStream_t hipxx_queue_ )
```

Construct a new HIPxxExecItem object.

Parameters

*grid_dim_*	
*block_dim_*	
*shared_↩ mem_*	
*hipxx_↩ queue_*	

4.13.3 Member Function Documentation

4.13.3.1 getBlock()

```
dim3 HIPxxExecItem::getBlock ( )
```

Get the Block object.

Returns

dim3

4.13.3.2 getGrid()

```
dim3 HIPxxExecItem::getGrid ( )
```

Get the Grid object.

Returns

dim3

4.13.3.3 getKernel()

```
HIPxxKernel* HIPxxExecItem::getKernel ( )
```

Get the Kernel object.

Returns

HIPxxKernel* Kernel to be executed

4.13.3.4 getQueue()

```
HIPxxQueue* HIPxxExecItem::getQueue ( )
```

Get the Queue object.

Returns

HIPxxQueue*

4.13.3.5 launch()

```
hipError_t HIPxxExecItem::launch (
            HIPxxKernel * Kernel )  [virtual]
```

Submit a kernel to the associated queue for execution. hipxx_queue must be set prior to this call.

Parameters

Kernel	kernel which is to be launched

Returns

hipError_t possible values: hipSuccess, hipErrorLaunchFailure

Reimplemented in HIPxxExecItemOpenCL.

4.13.3.6 launchByHostPtr()

```
hipError_t HIPxxExecItem::launchByHostPtr (
            const void * hostPtr )
```

Launch a kernel associated with a host function pointer. Looks up the HIPxxKernel associated with this pointer and calls launch()

Parameters

hostPtr	pointer to the host function

Returns

hipError_t possible values: hipSuccess, hipErrorLaunchFailure

4.13.3.7 setArg()

```
void HIPxxExecItem::setArg (
            const void * arg,
            size_t size,
            size_t offset )
```

Setup a single argument. gets called by hipSetupArgument calls to which are emitted by hip-clang.

Parameters

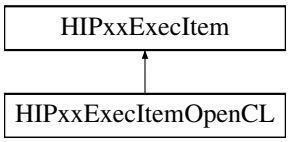
arg	
size	
offset	

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.14 HIPxxExecItemOpenCL Class Reference

Inheritance diagram for HIPxxExecItemOpenCL:



Public Member Functions

- virtual hipError_t launch (HIPxxKernel *hipxx_kernel) override

  Submit a kernel to the associated queue for execution. hipxx_queue must be set prior to this call.
- int **setup_all_args** (HIPxxKernelOpenCL *kernel)
- cl::Kernel * **get** ()

Public Attributes

- OCLFuncInfo **FuncInfo**

Additional Inherited Members

4.14.1 Member Function Documentation

4.14.1.1 launch()

```
hipError_t HIPxxExecItemOpenCL::launch (
            HIPxxKernel * Kernel )  [override], [virtual]
```

Submit a kernel to the associated queue for execution. hipxx_queue must be set prior to this call.

Parameters

Kernel	kernel which is to be launched

Returns

hipError_t possible values: hipSuccess, hipErrorLaunchFailure

Reimplemented from HIPxxExecItem.

The documentation for this class was generated from the following files:

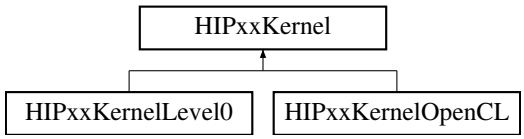
- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/HIPxxBackendOpenCL.cc

4.15 HIPxxKernel Class Reference

Contains information about the function on the host and device.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxKernel:



Public Member Functions

- std::string getName ()
  Get the Name object.
- const void * getHostPtr ()
  Get the associated host pointer to a host function.
- const void * getDevPtr ()
  Get the associated funciton pointer on the device.
- void setName (std::string host_f_name_)
  Get the Name object.
- void setHostPtr (const void *host_f_ptr_)
  Get the associated host pointer to a host function.
- void setDevPtr (const void *hev_f_ptr_)
  Get the associated funciton pointer on the device.

Protected Member Functions

- HIPxxKernel ()=default
  hidden default constructor. Only derived type constructor should be called.

Protected Attributes

- std::string host_f_name

  Name of the function.
- const void * host_f_ptr

  Pointer to the host function.
- const void * dev_f_ptr

  Pointer to the device function.

4.15.1 Detailed Description

Contains information about the function on the host and device.

4.15.2 Member Function Documentation

4.15.2.1 getDevPtr()

```
const void * HIPxxKernel::getDevPtr ( )
```

Get the associated funciton pointer on the device.

Returns

const void*

4.15.2.2 getHostPtr()

```
const void * HIPxxKernel::getHostPtr ( )
```

Get the associated host pointer to a host function.

Returns

const void*

4.15.2.3 getName()

```
std::string HIPxxKernel::getName ( )
```

Get the Name object.

Returns

std::string

4.15.2.4 setDevPtr()

```
void HIPxxKernel::setDevPtr (
            const void * hev_f_ptr_ )
```

Get the associated funciton pointer on the device.

Returns

const void*

4.15.2.5 setHostPtr()

```
void HIPxxKernel::setHostPtr (
            const void * host_f_ptr_ )
```

Get the associated host pointer to a host function.

Returns

const void*

4.15.2.6 setName()

```
void HIPxxKernel::setName (
            std::string host_f_name_ )
```

Get the Name object.

Returns

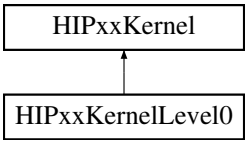
std::string

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.16 HIPxxKernelLevel0 Class Reference

Inheritance diagram for HIPxxKernelLevel0:



Public Member Functions

- **HIPxxKernelLevel0** (ze_kernel_handle_t _ze_kernel, std::string _funcName, const void *_host_ptr)
- ze_kernel_handle_t **get** ()

Protected Attributes

- ze_kernel_handle_t **ze_kernel**

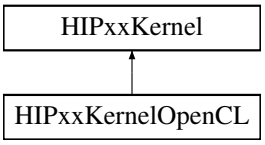
Additional Inherited Members

The documentation for this class was generated from the following file:

- src/backend/Level0/Level0Backend.hh

4.17 HIPxxKernelOpenCL Class Reference

Inheritance diagram for HIPxxKernelOpenCL:



Public Member Functions

- **HIPxxKernelOpenCL** (const cl::Kernel &&cl_kernel, OpenCLFunctionInfoMap &func_info_map)
- OCLFuncInfo * **get_func_info** () const
- std::string **get_name** ()
- cl::Kernel **get** () const
- size_t **getTotalArgSize** () const

Additional Inherited Members

The documentation for this class was generated from the following file:

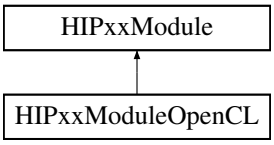
- src/backend/OpenCL/HIPxxBackendOpenCL.hh

4.18 HIPxxModule Class Reference

Module abstraction. Contains global variables and kernels. Can be extracted from FatBinary or loaded at runtime. OpenCL - ClProgram Level Zero - zeModule ROCclr - amd::Program CUDA - CUmodule.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxModule:



Public Member Functions

- ~HIPxxModule ()

  Destroy the HIPxxModule object.
- HIPxxModule (std::string *module_str)

  Construct a new HIPxxModule object. This constructor should be implemented by the derived class (specific backend implementation). Call to this constructor should result in a populated hipxx_kernels vector.
- HIPxxModule (std::string &&module_str)

  Construct a new HIPxxModule object using move semantics.
- void addKernel (HIPxxKernel *kernel)

  Add a HIPxxKernel to this module. During initialization when the FatBinary is consumed, a HIPxxModule is constructed for every device. SPIR-V kernels reside in this module. This method is called called via the constructor during this initialization phase. Modules can also be loaded from a file during runtime, however.
- void compileOnce (HIPxxDevice *hipxx_dev)

  Wrapper around compile() called via std::call_once.
- virtual void compile (HIPxxDevice *hipxx_dev)

  Kernel JIT compilation can be lazy. This is configured via Cmake LAZY_JIT option. If LAZY_JIT is set to true then this module won't be compiled until the first call to one of its kernels. If LAZY_JIT is set to false(default) then this method should be called in the constructor;.
- HIPxxDeviceVar * getGlobalVar (std::string name)

  Get the Global Var object A module, along with device kernels, can also contain global variables.
- HIPxxKernel * getKernel (std::string name)

  Get the Kernel object.
- std::vector< HIPxxKernel * > & getKernels ()

  Get the Kernels object.
- HIPxxKernel * getKernel (const void *host_f_ptr)

  Get the Kernel object.

Protected Member Functions

- HIPxxModule ()=default

  hidden default constuctor. Only derived type constructor should be called.

Protected Attributes

- std::mutex **mtx**
- std::vector< HIPxxDeviceVar * > **hipxx_vars**
- std::vector< HIPxxKernel * > **hipxx_kernels**
- std::string src

  Binary representation extracted from FatBinary.
- std::once_flag **compiled**

4.18.1 Detailed Description

Module abstraction. Contains global variables and kernels. Can be extracted from FatBinary or loaded at runtime. OpenCL - ClProgram Level Zero - zeModule ROCclr - amd::Program CUDA - CUmodule.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 HIPxxModule() [1/2]

```
HIPxxModule::HIPxxModule (
            std::string * module_str )
```

Construct a new HIPxxModule object. This constructor should be implemented by the derived class (specific backend implementation). Call to this constructor should result in a populated hipxx_kernels vector.

Parameters

module_str	string prepresenting the binary extracted from FatBinary

4.18.2.2 HIPxxModule() [2/2]

```
HIPxxModule::HIPxxModule (
            std::string && module_str )
```

Construct a new HIPxxModule object using move semantics.

Parameters

module_str	string from which to move resources

4.18.3 Member Function Documentation

4.18.3.1 addKernel()

```
void HIPxxModule::addKernel (
            HIPxxKernel * kernel )
```

Add a HIPxxKernel to this module. During initialization when the FatBinary is consumed, a HIPxxModule is constructed for every device. SPIR-V kernels reside in this module. This method is called called via the constructor during this initialization phase. Modules can also be loaded from a file during runtime, however.

Parameters

kernel	HIPxxKernel to be added to this module.

4.18.3.2 compile()

```
void HIPxxModule::compile (
            HIPxxDevice * hipxx_dev )  [virtual]
```

Kernel JIT compilation can be lazy. This is configured via Cmake LAZY_JIT option. If LAZY_JIT is set to true then this module won't be compiled until the first call to one of its kernels. If LAZY_JIT is set to false(default) then this method should be called in the constructor;.

This method should populate this modules hipxx_kernels vector. These kernels would have a name extracted from the kernel but no associated host function pointers.

Reimplemented in HIPxxModuleOpenCL.

4.18.3.3 compileOnce()

```
void HIPxxModule::compileOnce (
            HIPxxDevice * hipxx_dev )
```

Wrapper around compile() called via std::call_once.

Parameters

hipxx_dev	device for which to compile the kernels

4.18.3.4 getGlobalVar()

```
HIPxxDeviceVar * HIPxxModule::getGlobalVar (
            std::string name )
```

Get the Global Var object A module, along with device kernels, can also contain global variables.

Parameters

name	global variable name

Returns

HIPxxDeviceVar*

4.18.3.5 getKernel() [1/2]

```
HIPxxKernel * HIPxxModule::getKernel (
            const void * host_f_ptr )
```

Get the Kernel object.

Parameters

host_f_ptr	host-side function pointer

Returns

HIPxxKernel*

4.18.3.6 getKernel() [2/2]

```
HIPxxKernel * HIPxxModule::getKernel (
            std::string name )
```

Get the Kernel object.

Parameters

name	name of the corresponding host function

Returns

HIPxxKernel*

4.18.3.7 getKernels()

```
std::vector< HIPxxKernel * > & HIPxxModule::getKernels ( )
```

Get the Kernels object.

Returns

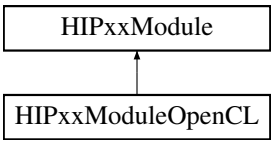
std::vector<HIPxxKernel*>&

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.19 HIPxxModuleOpenCL Class Reference

Inheritance diagram for HIPxxModuleOpenCL:



Public Member Functions

- virtual void compile (HIPxxDevice *hipxx_dev) override

  Kernel JIT compilation can be lazy. This is configured via Cmake LAZY_JIT option. If LAZY_JIT is set to true then this module won't be compiled until the first call to one of its kernels. If LAZY_JIT is set to false(default) then this method should be called in the constructor;.
- cl::Program & **get** ()

Protected Attributes

- cl::Program **program**

Additional Inherited Members

4.19.1 Member Function Documentation

4.19.1.1 compile()

```
void HIPxxModuleOpenCL::compile (
            HIPxxDevice * hipxx_dev )  [override], [virtual]
```

Kernel JIT compilation can be lazy. This is configured via Cmake LAZY_JIT option. If LAZY_JIT is set to true then this module won't be compiled until the first call to one of its kernels. If LAZY_JIT is set to false(default) then this method should be called in the constructor;.

This method should populate this modules hipxx_kernels vector. These kernels would have a name extracted from the kernel but no associated host function pointers.

Reimplemented from HIPxxModule.

The documentation for this class was generated from the following files:

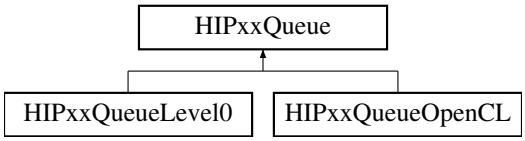
- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/HIPxxBackendOpenCL.cc

4.20 HIPxxQueue Class Reference

Queue class for submitting kernels to for execution.

`#include <HIPxxBackend.hh>`

Inheritance diagram for HIPxxQueue:



Public Member Functions

- HIPxxQueue (HIPxxDevice ∗hipxx_dev)

  Construct a new HIPxxQueue object.
- HIPxxQueue (HIPxxDevice ∗hipxx_dev, unsigned int flags)

  Construct a new HIPxxQueue object.
- HIPxxQueue (HIPxxDevice ∗hipxx_dev, unsigned int flags, int priority)

  Construct a new HIPxxQueue object.
- ∼HIPxxQueue ()

  Destroy the HIPxxQueue object.
- virtual hipError_t memCopy (void ∗dst, const void ∗src, size_t size)

  Blocking memory copy.
- virtual hipError_t memCopyAsync (void ∗dst, const void ∗src, size_t size)

  Non-blocking memory copy.
- virtual void memFill (void ∗dst, size_t size, const void ∗pattern, size_t pattern_size)

  Blocking memset.
- virtual void memFillAsync (void ∗dst, size_t size, const void ∗pattern, size_t pattern_size)

  Non-blocking mem set.
- virtual hipError_t launch (HIPxxExecItem ∗exec_item)

  Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.
- HIPxxDevice ∗ getDevice ()

  Get the Device obj.
- virtual void finish ()

  Wait for this queue to finish.
- bool query ()

  Check if the queue is still actively executing.
- int getPriorityRange (int lower_or_upper)

  Get the Priority Range object defining the bounds for hipStreamCreateWithPriority.
- bool enqueueBarrierForEvent (HIPxxEvent ∗e)

  Insert an event into this queue.
- unsigned int getFlags ()

  Get the Flags object with which this queue was created.
- int getPriority ()

  Get the Priority object with which this queue was created.
- bool addCallback (hipStreamCallback_t callback, void ∗userData)

Add a callback funciton to be called on the host after the specified stream is done.

- bool memPrefetch (const void *ptr, size_t count)

  Insert a memory prefetch.
- bool launchHostFunc (const void *hostFunction, dim3 numBlocks, dim3 dimBlocks, void **args, size_t sharedMemBytes)

  Launch a kernel on this queue given a host pointer and arguments.
- hipError_t launchWithKernelParams (dim3 grid, dim3 block, unsigned int sharedMemBytes, void **args, HIPxxKernel *kernel)
- hipError_t launchWithExtraParams (dim3 grid, dim3 block, unsigned int sharedMemBytes, void **extra, HIPxxKernel *kernel)

Protected Attributes

- std::mutex **mtx**
- int **priority**
- unsigned int **flags**
- HIPxxDevice * hipxx_device

  Device on which this queue will execute.
- HIPxxContext * hipxx_context

  Context to which device belongs to.

4.20.1 Detailed Description

Queue class for submitting kernels to for execution.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 HIPxxQueue() [1/3]

```
HIPxxQueue::HIPxxQueue (
            HIPxxDevice * hipxx_dev )
```

Construct a new HIPxxQueue object.

Parameters

hipxx_dev	

4.20.2.2 HIPxxQueue() [2/3]

```
HIPxxQueue::HIPxxQueue (
            HIPxxDevice * hipxx_dev,
            unsigned int flags )
```

Construct a new HIPxxQueue object.

Parameters

hipxx_dev	
flags	

4.20.2.3 HIPxxQueue() [3/3]

```
HIPxxQueue::HIPxxQueue (
            HIPxxDevice * hipxx_dev,
            unsigned int flags,
            int priority )
```

Construct a new HIPxxQueue object.

Parameters

hipxx_dev	
flags	
priority	

4.20.3 Member Function Documentation

4.20.3.1 addCallback()

```
bool HIPxxQueue::addCallback (
            hipStreamCallback_t callback,
            void * userData )
```

Add a callback funciton to be called on the host after the specified stream is done.

Parameters

callback	function pointer for a ballback function
userData	

Returns

true

false

4.20.3.2 enqueueBarrierForEvent()

```
bool HIPxxQueue::enqueueBarrierForEvent (
            HIPxxEvent * e )
```

Insert an event into this queue.

Parameters

e	

Returns

true

false

4.20.3.3 getDevice()

```
HIPxxDevice * HIPxxQueue::getDevice ( )
```

Get the Device obj.

Returns

HIPxxDevice*

4.20.3.4 getFlags()

```
unsigned int HIPxxQueue::getFlags ( )
```

Get the Flags object with which this queue was created.

Returns

unsigned int

4.20.3.5 getPriority()

```
int HIPxxQueue::getPriority ( )
```

Get the Priority object with which this queue was created.

Returns

int

4.20.3.6 getPriorityRange()

```
int HIPxxQueue::getPriorityRange (
            int lower_or_upper )
```

Get the Priority Range object defining the bounds for hipStreamCreateWithPriority.

Parameters

lower_or_upper	0 to get lower bound, 1 to get upper bound

Returns

int bound

4.20.3.7 launch()

```
virtual hipError_t HIPxxQueue::launch (
            HIPxxExecItem * exec_item )  [virtual]
```

Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.

Parameters

exec_item	

Returns

hipError_t

Reimplemented in HIPxxQueueOpenCL, and HIPxxQueueLevel0.

4.20.3.8 launchHostFunc()

```
bool HIPxxQueue::launchHostFunc (
            const void * hostFunction,
            dim3 numBlocks,
            dim3 dimBlocks,
            void ** args,
            size_t sharedMemBytes )
```

Launch a kernel on this queue given a host pointer and arguments.

Parameters

hostFunction	
numBlocks	
dimBlocks	
args	
sharedMemBytes	

Returns

true

false

4.20.3.9 launchWithExtraParams()

```
hipError_t HIPxxQueue::launchWithExtraParams (
            dim3 grid,
            dim3 block,
            unsigned int sharedMemBytes,
            void ** extra,
            HIPxxKernel * kernel )
```

Parameters

grid	
block	
sharedMemBytes	
extra	
kernel	

Returns

hipError_t

4.20.3.10 launchWithKernelParams()

```
hipError_t HIPxxQueue::launchWithKernelParams (
            dim3 grid,
            dim3 block,
            unsigned int sharedMemBytes,
            void ** args,
            HIPxxKernel * kernel )
```

Parameters

grid	
block	
sharedMemBytes	
args	
kernel	

Returns

hipError_t

4.20.3.11 memCopy()

```
virtual hipError_t HIPxxQueue::memCopy (
            void * dst,
            const void * src,
            size_t size )  [virtual]
```

Blocking memory copy.

Parameters

dst	Destination
src	Source
size	Transfer size

Returns

hipError_t

Reimplemented in HIPxxQueueOpenCL, and HIPxxQueueLevel0.

4.20.3.12 memCopyAsync()

```
virtual hipError_t HIPxxQueue::memCopyAsync (
            void * dst,
            const void * src,
            size_t size )  [virtual]
```

Non-blocking memory copy.

Parameters

dst	Destination
src	Source
size	Transfer size

Returns

hipError_t

4.20.3.13 memFill()

```
virtual void HIPxxQueue::memFill (
            void * dst,
```

```
            size_t size,
            const void * pattern,
            size_t pattern_size )  [virtual]
```

Blocking memset.

Parameters

dst	
size	
pattern	
pattern_size	

4.20.3.14 memFillAsync()

```
virtual void HIPxxQueue::memFillAsync (
            void * dst,
            size_t size,
            const void * pattern,
            size_t pattern_size )  [virtual]
```

Non-blocking mem set.

Parameters

dst	
size	
pattern	
pattern_size	

4.20.3.15 memPrefetch()

```
bool HIPxxQueue::memPrefetch (
            const void * ptr,
            size_t count )
```

Insert a memory prefetch.

Parameters

ptr	
count	

Returns

true

false

4.20.3.16 query()

```
bool HIPxxQueue::query ( )
```

Check if the queue is still actively executing.

Returns

true

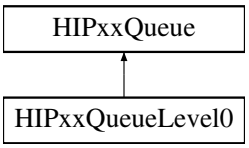
false

The documentation for this class was generated from the following files:

- src/HIPxxBackend.hh
- src/HIPxxBackend.cc

4.21 HIPxxQueueLevel0 Class Reference

Inheritance diagram for HIPxxQueueLevel0:



Public Member Functions

- **HIPxxQueueLevel0** (HIPxxDeviceLevel0 *hixx_dev_)
- virtual hipError_t launch (HIPxxExecItem *exec_item) override

  Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.
- ze_command_queue_handle_t **get** ()
- virtual hipError_t memCopy (void *dst, const void *src, size_t size) override

  Blocking memory copy.

Protected Attributes

- ze_command_queue_handle_t **ze_q**
- ze_context_handle_t **ze_ctx**
- ze_device_handle_t **ze_dev**

4.21.1 Member Function Documentation

4.21.1.1 launch()

```
virtual hipError_t HIPxxQueueLevel0::launch (
            HIPxxExecItem * exec_item )  [inline], [override], [virtual]
```

Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.

Parameters

exec_item	

Returns

hipError_t

Reimplemented from HIPxxQueue.

4.21.1.2 memCopy()

```
hipError_t HIPxxQueueLevel0::memCopy (
            void * dst,
            const void * src,
            size_t size )  [override], [virtual]
```

Blocking memory copy.

Parameters

dst	Destination
src	Source
size	Transfer size

Returns

hipError_t

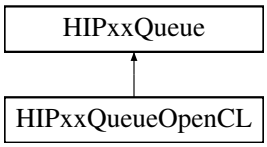
Reimplemented from HIPxxQueue.

The documentation for this class was generated from the following files:

- src/backend/Level0/Level0Backend.hh
- src/backend/Level0/Level0Backend.cc

4.22 HIPxxQueueOpenCL Class Reference

Inheritance diagram for HIPxxQueueOpenCL:



Public Member Functions

- **HIPxxQueueOpenCL** (const HIPxxQueueOpenCL &)=delete
- **HIPxxQueueOpenCL** (HIPxxDevice *hipxx_device)
- virtual hipError_t launch (HIPxxExecItem *exec_item) override

  Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.
- virtual void finish () override

  Wait for this queue to finish.
- virtual hipError_t memCopy (void *dst, const void *src, size_t size) override

  Blocking memory copy.
- cl::CommandQueue * **get** ()

Protected Attributes

- cl::Context * **cl_ctx**
- cl::Device * **cl_dev**
- cl::CommandQueue * **cl_q**

4.22.1 Member Function Documentation

4.22.1.1 launch()

```
hipError_t HIPxxQueueOpenCL::launch (
            HIPxxExecItem * exec_item )  [override], [virtual]
```

Submit a HIPxxExecItem to this queue for execution. HIPxxExecItem needs to be complete - contain the kernel and arguments.

Parameters

exec_item	

Returns

hipError_t

Reimplemented from HIPxxQueue.

4.22.1.2 memCopy()

```
hipError_t HIPxxQueueOpenCL::memCopy (
            void * dst,
            const void * src,
            size_t size )  [override], [virtual]
```

Blocking memory copy.

Parameters

dst	Destination
src	Source
size	Transfer size

Returns

hipError_t

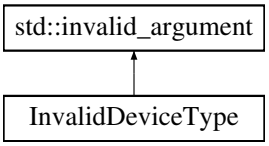
Reimplemented from HIPxxQueue.

The documentation for this class was generated from the following files:

- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/HIPxxBackendOpenCL.cc

4.23 InvalidDeviceType Class Reference

Inheritance diagram for InvalidDeviceType:

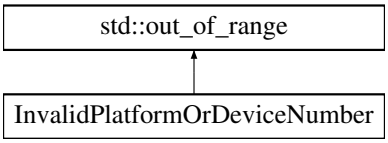


The documentation for this class was generated from the following file:

- src/backend/OpenCL/exceptions.hh

4.24 InvalidPlatformOrDeviceNumber Class Reference

Inheritance diagram for InvalidPlatformOrDeviceNumber:



The documentation for this class was generated from the following file:

- src/backend/OpenCL/exceptions.hh

4.25 OCLArgTypeInfo Struct Reference

Public Attributes

- OCLType **type**
- OCLSpace **space**
- size_t **size**

The documentation for this struct was generated from the following file:

- src/common.hh

4.26 OCLFuncInfo Struct Reference

Public Attributes

- std::vector< OCLArgTypeInfo > **ArgTypeInfo**
- OCLArgTypeInfo **retTypeInfo**

The documentation for this struct was generated from the following file:

- src/common.hh

4.27 SPIRVinst Class Reference

Public Member Functions

- **SPIRVinst** (int32_t ∗stream)
- bool **isKernelCapab** () const
- bool **isExtIntOpenCL** () const
- bool **isMemModelOpenCL** () const
- size_t **getPointerSize** () const
- bool **isLangOpenCL** () const
- bool **isEntryPoint** ()
- int32_t **entryPointID** ()
- std::string && **entryPointName** ()
- size_t **size** () const
- spv::Op **getOpcode** () const
- int32_t **getFunctionID** () const
- int32_t **getFunctionTypeID** () const
- int32_t **getFunctionRetType** () const
- bool **isType** () const
- int32_t **getTypeID** () const
- bool **isFunctionType** () const
- bool **isFunction** () const
- SPIRVtype ∗ **decodeType** (SPIRTypeMap &typeMap, size_t pointerSize)
- OCLFuncInfo ∗ **decodeFunctionType** (SPIRTypeMap &typeMap, size_t pointerSize)

The documentation for this class was generated from the following file:

- src/spirv.cc

4.28 SPIRVmodule Class Reference

Public Member Functions

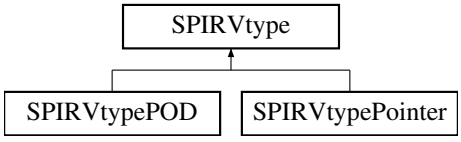
- bool **valid** ()
- bool **parseSPIRV** (int32_t ∗stream, size_t numWords)
- bool **fillModuleInfo** (OpenCLFunctionInfoMap &moduleMap)

The documentation for this class was generated from the following file:

- src/spirv.cc

4.29 SPIRVtype Class Reference

Inheritance diagram for SPIRVtype:



Public Member Functions

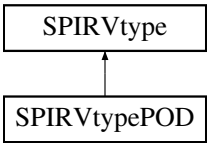
- **SPIRVtype** (size_t s)
- size_t **size** ()
- virtual OCLType **ocltype** ()=0
- virtual OCLSpace **getAS** ()

The documentation for this class was generated from the following file:

- src/spirv.cc

4.30 SPIRVtypePOD Class Reference

Inheritance diagram for SPIRVtypePOD:



Public Member Functions

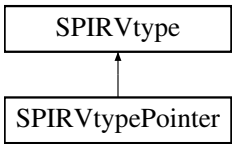
- **SPIRVtypePOD** (int32_t id, size_t size)
- virtual OCLType **ocltype** () override

The documentation for this class was generated from the following file:

- src/spirv.cc

4.31 SPIRVtypePointer Class Reference

Inheritance diagram for SPIRVtypePointer:



Public Member Functions

- **SPIRVtypePointer** (int32_t id, int32_t stor_class, size_t pointerSize)
- virtual OCLType **ocltype** () override
- OCLSpace **getAS** () override

The documentation for this class was generated from the following file:

- src/spirv.cc

4.32 SVMemoryRegion Class Reference

Public Member Functions

- void **init** (cl::Context &C)
- SVMemoryRegion & **operator=** (SVMemoryRegion &&rhs)
- void * **allocate** (cl::Context ctx, size_t size)
- bool **free** (void *p, size_t *size)
- bool **hasPointer** (const void *p)
- bool **pointerSize** (void *ptr, size_t *size)
- bool **pointerInfo** (void *ptr, void **pbase, size_t *psize)
- int **memCopy** (void *dst, const void *src, size_t size, cl::CommandQueue &queue)
- int **memFill** (void *dst, size_t size, const void *pattern, size_t patt_size, cl::CommandQueue &queue)
- void **clear** ()

The documentation for this class was generated from the following files:

- src/backend/OpenCL/HIPxxBackendOpenCL.hh
- src/backend/OpenCL/SVMemoryRegion.cc

Chapter 5

File Documentation

5.1 src/HIPxxBackend.hh File Reference

HIPxxBackend class definition. HIPxx backends are to inherit from this base class and override desired virtual functions. Overrides for this class are expected to be minimal with primary overrides being done on lower-level classes such as HIPxxContext consturctors, etc.

```
#include <algorithm>
#include <iostream>
#include <map>
#include <mutex>
#include <string>
#include <vector>
#include <stack>
#include "spirv.hh"
#include "include/hip/hip.hh"
#include "HIPxxDriver.hh"
#include "logging.hh"
#include "macros.hh"
```

Classes

- class HIPxxDeviceVar
- class HIPxxEvent
- class HIPxxModule

  Module abstraction. Contains global variables and kernels. Can be extracted from FatBinary or loaded at runtime. OpenCL - ClProgram Level Zero - zeModule ROCclr - amd::Program CUDA - CUmodule.
- class HIPxxKernel

  Contains information about the function on the host and device.
- class HIPxxExecItem

  Contains kernel arguments and a queue on which to execute. Prior to kernel launch, the arguments are setup via HIPxxBackend::configureCall(). Because of this, we get the kernel last so the kernel so the launch() takes a kernel argument as opposed to queue receiving a HIPxxExecItem containing the kernel and arguments.
- class HIPxxDevice

  Compute device class.
- class HIPxxContext

Context class Contexts contain execution queues and are created on top of a single or multiple devices. Provides for creation of additional queues, events, and interaction with devices.

- class HIPxxBackend

  Primary object to interact with the backend.

- class HIPxxQueue

  Queue class for submitting kernels to for execution.

Enumerations

- enum class **HIPxxMemoryType** : unsigned { **Host** = 0 , **Device** = 1 , **Shared** = 2 }
- enum class **HIPxxEventType** : unsigned { **Default** = hipEventDefault , **BlockingSync** = hipEventBlocking↩ Sync , **DisableTiming** = hipEventDisableTiming , **Interprocess** = hipEventInterprocess }

5.1.1 Detailed Description

HIPxxBackend class definition. HIPxx backends are to inherit from this base class and override desired virtual functions. Overrides for this class are expected to be minimal with primary overrides being done on lower-level classes such as HIPxxContext consturctors, etc.

Author

Paulius Velesko (pvelesko@gmail.com)

Version

0.1

Date

2021-08-19

Copyright

5.2 src/HIPxxDriver.cc File Reference

Definitions of extern declared functions and objects in HIPxxDriver.hh Initializing the HIPxx runtime with backend selection through HIPXX_BE environment variable.

```
#include "HIPxxDriver.hh"
#include <string>
#include "backend/backends.hh"
```

Functions

- std::string **read_env_var** (std::string ENV_VAR)
- std::string **read_backend_selection** ()
- void **read_env_vars** (std::string &HIPxxPlatformStr, std::string &HIPxxDeviceTypeStr, std::string &HIPxx↩ DeviceStr)
- void HIPxxInitializeCallOnce (std::string BE)

  Singleton backend initialization function called via std::call_once.
- void HIPxxInitialize (std::string BE)

  Singleton backend initialization function outer wrapper.
- void HIPxxUninitializeCallOnce ()

  Singleton backend uninitialization function called via std::call_once.
- void HIPxxUninitialize ()

  Singleton backend initialization function outer wrapper.

Variables

- std::once_flag initialized

  Singleton backend initialization flag.
- std::once_flag **uninitialized**
- HIPxxBackend * Backend

  Global Backend pointer through which backend-specific operations are performed.

5.2.1 Detailed Description

Definitions of extern declared functions and objects in HIPxxDriver.hh Initializing the HIPxx runtime with backend selection through HIPXX_BE environment variable.

Author

Paulius Velesko (pvelesko@gmail.com)

Version

0.1

Date

2021-08-19

Copyright

Copyright (c) 2021

5.3 src/HIPxxDriver.hh File Reference

Header defining global HIPxx classes and functions such as HIPxxBackend type pointer Backend which gets initialized at the start of execution.

```
#include <iostream>
#include <mutex>
#include "HIPxxBackend.hh"
```

Functions

- void HIPxxInitialize (std::string BE="")

  Singleton backend initialization function outer wrapper.
- void HIPxxUninitialize ()

  Singleton backend initialization function outer wrapper.
- void HIPxxInitializeCallOnce (std::string BE="")

  Singleton backend initialization function called via std::call_once.
- void HIPxxUninitializeCallOnce ()

  Singleton backend uninitialization function called via std::call_once.
- std::string **read_env_var** (std::string ENV_VAR)
- std::string **read_backend_selection** ()

Variables

- HIPxxBackend * Backend

  Global Backend pointer through which backend-specific operations are performed.
- std::once_flag initialized

  Singleton backend initialization flag.
- std::once_flag **uninitialized**

5.3.1 Detailed Description

Header defining global HIPxx classes and functions such as HIPxxBackend type pointer Backend which gets initialized at the start of execution.

Author

Paulius Velesko (pvelesko@gmail.com)

Version

0.1

Date

2021-08-19

Copyright

Copyright (c) 2021

Index