
cudawrappers

Release 0.6.0

cudawrappers developers

Apr 26, 2024

API DOCUMENTATION

| | | |
|-----------|--------------------------------|-----------|
| 1 | Requirements | 3 |
| 2 | Usage | 5 |
| 2.1 | Usage examples | 5 |
| 3 | Used by | 7 |
| 4 | Alternatives | 9 |
| 4.1 | cuda-api-wrappers | 9 |
| 4.2 | cuda-wrapper | 9 |
| 4.3 | CudaPlusPlus | 9 |
| 5 | Contributing | 11 |
| 6 | Developer documentation | 13 |
| 7 | Class Hierarchy | 15 |
| 8 | File Hierarchy | 17 |
| 9 | Full API | 19 |
| 9.1 | Namespaces | 19 |
| 9.2 | Classes and Structs | 21 |
| 9.3 | Functions | 35 |
| 10 | Indices and tables | 37 |
| | Index | 39 |

This library is a C++ wrapper for the Nvidia C libraries (e.g. CUDA driver, nvrtc, cuFFT etc.). The main purposes are:

1. *easier resource management*, leading to *lower risk of programming errors*;
2. *better fault handling* (through exceptions);
3. *more compact user code*.

Originally, the API enforced RAII to even further reduce the risk of faulty code, but enforcing RAII and compatibility with (unmanaged) objects obtained outside this API are mutually exclusive.

REQUIREMENTS

| Software | Minimum version |
|----------|----------------------|
| CUDA | 10.0 or later |
| CMake | 3.17 or later |
| gcc | 9.3 or later |
| OS | Linux distro (amd64) |

| Hardware | Type |
|------------------|-------------------------------|
| GPU architecture | NVIDIA PASCAL or newer |

USAGE

We use CMake in this project, so you can clone and build this library with the following steps:

```
git clone https://github.com/nlesc-recruit/cudawrappers
cd cudawrappers
cmake -S . -B build
make -C build
```

This command will create a build directory. Cudawrappers is header only, so no library objects are being built. For more details on the building requirements and on testing, check the *developer documentation*.

To install to `~/local`, use

```
git clone https://github.com/nlesc-recruit/cudawrappers
cd cudawrappers
cmake -DCMAKE_INSTALL_PREFIX=$HOME/.local -S . -B build
make -C build
make -C build install
```

2.1 Usage examples

You can include the cudawrappers library in your own projects in various ways. We have created a few repositories with example setups to get you started:

1. [cudawrappers-usage-example-git-submodules](#) Example project that uses the cudawrappers library as a dependency by using git submodules on its source tree.
2. [cudawrappers-usage-example-locally-installed](#) Example project that uses the cudawrappers library as a dependency by having it locally installed.
3. [cudawrappers-usage-example-cmake-pull](#) Example project that uses the cudawrappers library as a dependency by having cmake pull it in from github.

USED BY

This section aims to provide an overview of projects that use this repo's library (or something very similar), e.g. through git submodules or by including copies of this library in their source tree:

1. <https://git.astron.nl/RD/dedisp/>
2. <https://git.astron.nl/RD/idg>
3. <https://git.astron.nl/RD/tensor-core-correlator>

ALTERNATIVES

This section provides an overview of similar tools in this space, and how they are different.

4.1 `cuda-api-wrappers`

- Aims to provide wrappers for the CUDA runtime API
- Development has slowed a bit recently
- Has 1 or 2 main developers
- Has gained quite a bit of attention (e.g. 440 stars; 57 forks)

The project is planning to support more of the Driver API (for fine-grained control of CUDA devices) and NVRTC API (for runtime compilation of kernels); there is a release candidate ([v0.5.0-rc1](#)). It doesn't provide support for cuFFT and cuBLAS though.

4.2 `cuda-wrapper`

- Aims to provide a C++ wrapper for the CUDA Driver and Runtime APIs

4.3 `CudaPlusPlus`

- Aims to provide a C++ wrapper for the CUDA Driver API
- Project appears inactive

CONTRIBUTING

See [CONTRIBUTING.md](#) for a guide on how to contribute.

DEVELOPER DOCUMENTATION

See [README.dev.md](#) for documentation on setting up your development environment.

CLASS HIERARCHY

FILE HIERARCHY

9.1 Namespaces

9.1.1 Namespace cu

Contents

- *Classes*
- *Functions*

Classes

- *Struct Device::CUdeviceArg*
- *Class Array*
- *Class Context*
- *Class Device*
- *Class DeviceMemory*
- *Class Error*
- *Class Event*
- *Class Function*
- *Class HostMemory*
- *Class Module*
- *Class Stream*
- *Template Class Wrapper*

Functions

- *Function cu::checkCudaCall*
- *Function cu::driverGetVersion*
- *Function cu::init*
- *Function cu::memcpyDtoH*
- *Function cu::memcpyHtoD*

9.1.2 Namespace cufft

Contents

- *Classes*

Classes

- *Class Error*
- *Class FFT*
- *Template Class FFT1D*
- *Template Class FFT2D*

9.1.3 Namespace nvrtc

Contents

- *Classes*
- *Functions*

Classes

- *Class Error*
- *Class Program*

Functions

- *Function nvrtc::checkNvrtcCall*

9.1.4 Namespace nvtx

Contents

- *Classes*

Classes

- *Class Marker*

9.1.5 Namespace std

9.2 Classes and Structs

9.2.1 Struct Device::CUdeviceArg

- Defined in file_include_cudaWrappers_cu.hpp

Nested Relationships

This struct is a nested type of *Class Device*.

Struct Documentation

struct **CUdeviceArg**

9.2.2 Class Array

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUarray > (*Template Class Wrapper*)

Class Documentation

class **Array** : public cu::Wrapper<CUarray>

Public Functions

inline **Array**(unsigned width, CUarray_format format, unsigned numChannels)

inline **Array**(unsigned width, unsigned height, CUarray_format format, unsigned numChannels)

inline **Array**(unsigned width, unsigned height, unsigned depth, CUarray_format format, unsigned numChannels)

inline explicit **Array**(CUarray &array)

9.2.3 Class Context

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUcontext > (*Template Class Wrapper*)

Class Documentation

class **Context** : public cu::Wrapper<CUcontext>

Public Functions

inline **Context**(int flags, *Device* &device)

inline explicit **Context**(CUcontext context)

inline unsigned **getApiVersion**() const

inline void **setCurrent**() const

inline void **pushCurrent**()

inline void **setSharedMemConfig**(CUsharedconfig config)

inline size_t **getFreeMemory**() const

inline size_t **getTotalMemory**() const

Public Static Functions

```
static inline CUfunc_cache getCacheConfig()

static inline void setCacheConfig(CUfunc_cache config)

static inline Context getCurrent()

static inline Context popCurrent()

static inline Device getDevice()

static inline size_t getLimit(CUlimit limit)

template<CUlimit limit>
static inline size_t getLimit()

static inline void setLimit(CUlimit limit, size_t value)

template<CUlimit limit>
static inline void setLimit(size_t value)

static inline void synchronize()
```

9.2.4 Class Device

- Defined in file_include_cudawrappers_cu.hpp

Nested Relationships

Nested Types

- *Struct Device::CUdeviceArg*

Inheritance Relationships

Base Type

- public cu::Wrapper< CUdevice > (*Template Class Wrapper*)

Class Documentation

```
class Device : public cu::Wrapper<CUdevice>
```

Public Functions

```

inline explicit Device(int ordinal)

inline Device(CUdeviceArg, CUdevice device)

inline int getAttribute(CUdevice_attribute attribute) const

template<CUdevice_attribute attribute>
inline int getAttribute() const

inline std::string getName() const

inline size_t totalMem() const

inline std::pair<unsigned, bool> primaryCtxGetState() const

inline void primaryCtxReset()

Context primaryCtxRetain()

inline void primaryCtxSetFlags(unsigned flags)

```

Public Static Functions

```

static inline int getCount()

struct CUdeviceArg

```

9.2.5 Class DeviceMemory

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUdeviceptr > (*Template Class Wrapper*)

Class Documentation

```

class DeviceMemory : public cu::Wrapper<CUdeviceptr>

```

Public Functions

```

inline explicit DeviceMemory(size_t size, CUmemorytype type = CU_MEMORYTYPE_DEVICE, unsigned
                             int flags = 0)

inline explicit DeviceMemory(CUdeviceptr ptr)

inline explicit DeviceMemory(CUdeviceptr ptr, size_t size)

inline explicit DeviceMemory(const HostMemory &hostMemory)

inline void zero(size_t size)

inline const void *parameter() const

template<typename T>
inline operator T*()

inline size_t size() const

```

9.2.6 Class Error

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public std::exception

Class Documentation

class **Error** : public std::exception

Public Functions

```

inline explicit Error(CUresult result)

inline const char *what() const noexcept

inline operator CUresult() const

```

9.2.7 Class Event

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUevent > (*Template Class Wrapper*)

Class Documentation

class **Event** : public cu::Wrapper<CUevent>

Public Functions

inline explicit **Event**(unsigned int flags = CU_EVENT_DEFAULT)

inline explicit **Event**(CUevent &event)

inline float **elapsedTime**(const *Event* &start) const

inline void **query**() const

inline void **record**()

inline void **record**(*Stream*&)

inline void **synchronize**()

9.2.8 Class Function

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUfunction > (*Template Class Wrapper*)

Class Documentation

class **Function** : public cu::Wrapper<CUfunction>

Public Functions

```

inline Function(const Module &module, const char *name)

inline explicit Function(CUfunction &function)

inline int getAttribute(CUfunction_attribute attribute) const

inline void setAttribute(CUfunction_attribute attribute, int value)

inline int occupancyMaxActiveBlocksPerMultiprocessor(int blockSize, size_t dynamicSMemSize)

inline void setCacheConfig(CUfunc_cache config)

inline const char *name() const

```

9.2.9 Class HostMemory

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< void * > (*Template Class Wrapper*)

Class Documentation

```
class HostMemory : public cu::Wrapper<void*>
```

Public Functions

```

inline explicit HostMemory(size_t size, unsigned int flags = 0)

inline explicit HostMemory(void *ptr, size_t size, unsigned int flags = 0)

template<typename T>
inline operator T*()

inline size_t size() const

```

9.2.10 Class Module

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- `public cu::Wrapper< CUmodule >` (*Template Class Wrapper*)

Class Documentation

class **Module** : public cu::Wrapper<CUmodule>

Public Types

typedef std::map<CUjit_option, void*> **optionmap_t**

Public Functions

inline explicit **Module**(const char *file_name)

inline explicit **Module**(const void *data)

inline explicit **Module**(const void *image, *Module*::optionmap_t &options)

inline explicit **Module**(CUmodule &module)

inline CUdeviceptr **getGlobal**(const char *name) const

9.2.11 Class Stream

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- `public cu::Wrapper< CUstream >` (*Template Class Wrapper*)

Class Documentation

class **Stream** : public cu::Wrapper<CUstream>

Public Functions

```

inline explicit Stream(unsigned int flags = CU_STREAM_DEFAULT)

inline explicit Stream(CUstream stream)

inline DeviceMemory memAllocAsync(size_t size)

inline void memFreeAsync(DeviceMemory &devMem)

inline void memcpyHtoHAsync(void *dstPtr, const void *srcPtr, size_t size)

inline void memcpyHtoDAsync(DeviceMemory &devPtr, const void *hostPtr, size_t size)

inline void memcpyHtoDAsync(CUdeviceptr devPtr, const void *hostPtr, size_t size)

inline void memcpyDtoHAsync(void *hostPtr, const DeviceMemory &devPtr, size_t size)

inline void memcpyDtoHAsync(void *hostPtr, CUdeviceptr devPtr, size_t size)

inline void memcpyDtoDAsync(DeviceMemory &dstPtr, DeviceMemory &srcPtr, size_t size)

inline void memPrefetchAsync(DeviceMemory &devPtr, size_t size)

inline void memPrefetchAsync(DeviceMemory &devPtr, size_t size, Device &dstDevice)

inline void zero(DeviceMemory &devPtr, size_t size)

inline void launchKernel(Function &function, unsigned gridX, unsigned gridY, unsigned gridZ, unsigned
    blockX, unsigned blockY, unsigned blockZ, unsigned sharedMemBytes, const
    std::vector<const void*> &parameters)

inline void query()

inline void synchronize()

inline void wait(Event &event)

inline void addCallback(CUstreamCallback callback, void *userData, unsigned int flags = 0)

inline void record(Event &event)

inline void batchMemOp(unsigned count, CUstreamBatchMemOpParams *paramArray, unsigned flags)

inline void waitValue32(CUdeviceptr addr, cuuint32_t value, unsigned flags) const

inline void writeValue32(CUdeviceptr addr, cuuint32_t value, unsigned flags)

inline Context getContext() const

```

9.2.12 Template Class Wrapper

- Defined in file_include_cudaWrappers_cu.hpp

Class Documentation

template<typename **T**>

class **Wrapper**

Public Functions

inline **operator** **T**() const

inline **operator** **T**()

inline bool **operator**==(const *Wrapper*<**T**> &other)

inline bool **operator**!=(const *Wrapper*<**T**> &other)

Protected Functions

Wrapper() = default

inline **Wrapper**(const *Wrapper*<**T**> &other)

inline **Wrapper**(*Wrapper*<**T**> &&other)

inline explicit **Wrapper**(**T** &obj)

Protected Attributes

T _obj = { }

std::shared_ptr<**T**> **manager**

9.2.13 Class Error

- Defined in file_include_cudawrappers_cufft.hpp

Inheritance Relationships

Base Type

- public std::exception

Class Documentation

class **Error** : public std::exception

Public Functions

inline explicit **Error**(cufftResult result)

inline const char ***what**() const noexcept override

inline **operator** cufftResult() const

9.2.14 Class FFT

- Defined in file_include_cudaWrappers_cufft.hpp

Inheritance Relationships

Derived Types

- public cufft::FFT1D< T > (*Template Class FFT1D*)
- public cufft::FFT2D< T > (*Template Class FFT2D*)

Class Documentation

class **FFT**

Subclassed by *cufft::FFT1D< T >*, *cufft::FFT2D< T >*

Public Functions

FFT() = default

FFT &**operator**=(*FFT* &) = delete

FFT(*FFT* &) = delete

inline *FFT* &**operator**=(*FFT* &&other) noexcept

inline **FFT**(*FFT* &&other) noexcept

inline ~**FFT**()

inline void **setStream**(cu::Stream &stream)

inline void **execute**(cu::DeviceMemory &in, cu::DeviceMemory &out, int direction)

Protected Functions

```
inline void checkCuFFTCall(cufftResult result)

inline cufftHandle *plan()
```

9.2.15 Template Class FFT1D

- Defined in file_include_cudawrappers_cufft.hpp

Inheritance Relationships

Base Type

- public cufft::FFT (*Class FFT*)

Class Documentation

```
template<cudaDataType_t T>
class FFT1D : public cufft::FFT
```

Public Functions

```
FFT1D(int nx) = delete
FFT1D(int nx, int batch) = delete
FFT1D(int nx, int batch)
FFT1D(int nx)
FFT1D(int nx, int batch)
FFT1D(int nx)
```

9.2.16 Template Class FFT2D

- Defined in file_include_cudawrappers_cufft.hpp

Inheritance Relationships

Base Type

- public cufft::FFT (*Class FFT*)

Class Documentation

template<cudaDataType_t T>

class **FFT2D** : public cufft::*FFT*

Public Functions

FFT2D(int nx, int ny) = delete

FFT2D(int nx, int ny, int stride, int dist, int batch) = delete

FFT2D(int nx, int ny)

FFT2D(int nx, int ny, int stride, int dist, int batch)

FFT2D(int nx, int ny, int stride, int dist, int batch)

FFT2D(int nx, int ny)

9.2.17 Class Error

- Defined in file_include_cudaWrappers_nvrtc.hpp

Inheritance Relationships

Base Type

- public std::exception

Class Documentation

class **Error** : public std::exception

Public Functions

inline explicit **Error**(nvrtcResult result)

inline const char ***what**() const noexcept

inline **operator** **nvcrtResult**() const

9.2.18 Class Program

- Defined in file_include_cudawrappers_nvrtc.hpp

Class Documentation

class **Program**

Public Functions

```
inline Program(const std::string &src, const std::string &name, const std::vector<std::string> &headers =
    std::vector<std::string>(), const std::vector<std::string> &includeNames =
    std::vector<std::string>())
```

```
inline explicit Program(const std::string &filename)
```

```
inline ~Program()
```

```
inline void compile(const std::vector<std::string> &options)
```

```
inline std::string getPTX()
```

```
inline std::string getLog()
```

9.2.19 Class Marker

- Defined in file_include_cudawrappers_nvtx.hpp

Class Documentation

class **Marker**

Public Types

```
enum Color
```

Values:

```
enumerator red
```

```
enumerator green
```

```
enumerator blue
```

```
enumerator yellow
```

```
enumerator black
```

Public Functions

inline explicit **Marker**(const char *message, unsigned color = *Color::green*)

inline **Marker**(const char *message, *Color* color)

inline void **start**()

inline void **end**()

9.3 Functions

9.3.1 Function `_cudaGetErrorEnum`

- Defined in file_include_cudawrappers_cufft.hpp

Function Documentation

static const char ***_cudaGetErrorEnum**(cufftResult error)

9.3.2 Function `cu::checkCudaCall`

- Defined in file_include_cudawrappers_cu.hpp

Function Documentation

inline void **cu::checkCudaCall**(CUresult result)

9.3.3 Function `cu::driverGetVersion`

- Defined in file_include_cudawrappers_cu.hpp

Function Documentation

inline int **cu::driverGetVersion**()

9.3.4 Function `cu::init`

- Defined in file_include_cudawrappers_cu.hpp

Function Documentation

inline void **cu::init**(unsigned flags = 0)

9.3.5 Function **cu::memcpyDtoH**

- Defined in file_include_cudawrappers_cu.hpp

Function Documentation

inline void **cu::memcpyDtoH**(void *dst, CUdeviceptr src, size_t size)

9.3.6 Function **cu::memcpyHtoD**

- Defined in file_include_cudawrappers_cu.hpp

Function Documentation

inline void **cu::memcpyHtoD**(CUdeviceptr dst, const void *src, size_t size)

9.3.7 Function **nVRTC::checkNVRTCcall**

- Defined in file_include_cudawrappers_nVRTC.hpp

Function Documentation

inline void **nVRTC::checkNVRTCcall**(nVRTCResult result)

INDICES AND TABLES

- `genindex`
- `search`

Symbols

`_cudaGetErrorEnum` (C++ function), 35

C

`cu::Array` (C++ class), 22

`cu::Array::Array` (C++ function), 22

`cu::checkCudaCall` (C++ function), 35

`cu::Context` (C++ class), 22

`cu::Context::Context` (C++ function), 22

`cu::Context::getApiVersion` (C++ function), 22

`cu::Context::getCacheConfig` (C++ function), 23

`cu::Context::getCurrent` (C++ function), 23

`cu::Context::getDevice` (C++ function), 23

`cu::Context::getFreeMemory` (C++ function), 22

`cu::Context::getLimit` (C++ function), 23

`cu::Context::getTotalMemory` (C++ function), 22

`cu::Context::popCurrent` (C++ function), 23

`cu::Context::pushCurrent` (C++ function), 22

`cu::Context::setCacheConfig` (C++ function), 23

`cu::Context::setCurrent` (C++ function), 22

`cu::Context::setLimit` (C++ function), 23

`cu::Context::setSharedMemConfig` (C++ function),
22

`cu::Context::synchronize` (C++ function), 23

`cu::Device` (C++ class), 23

`cu::Device::CUdeviceArg` (C++ struct), 21, 24

`cu::Device::Device` (C++ function), 24

`cu::Device::getAttribute` (C++ function), 24

`cu::Device::getCount` (C++ function), 24

`cu::Device::getName` (C++ function), 24

`cu::Device::primaryCtxGetState` (C++ function),
24

`cu::Device::primaryCtxReset` (C++ function), 24

`cu::Device::primaryCtxRetain` (C++ function), 24

`cu::Device::primaryCtxSetFlags` (C++ function),
24

`cu::Device::totalMem` (C++ function), 24

`cu::DeviceMemory` (C++ class), 24

`cu::DeviceMemory::DeviceMemory` (C++ function),
25

`cu::DeviceMemory::operator T*` (C++ function), 25

`cu::DeviceMemory::parameter` (C++ function), 25

`cu::DeviceMemory::size` (C++ function), 25

`cu::DeviceMemory::zero` (C++ function), 25

`cu::driverGetVersion` (C++ function), 35

`cu::Error` (C++ class), 25

`cu::Error::Error` (C++ function), 25

`cu::Error::operator CUresult` (C++ function), 25

`cu::Error::what` (C++ function), 25

`cu::Event` (C++ class), 26

`cu::Event::elapsedTime` (C++ function), 26

`cu::Event::Event` (C++ function), 26

`cu::Event::query` (C++ function), 26

`cu::Event::record` (C++ function), 26

`cu::Event::synchronize` (C++ function), 26

`cu::Function` (C++ class), 26

`cu::Function::Function` (C++ function), 27

`cu::Function::getAttribute` (C++ function), 27

`cu::Function::name` (C++ function), 27

`cu::Function::occupancyMaxActiveBlocksPerMultiprocessor`
(C++ function), 27

`cu::Function::setAttribute` (C++ function), 27

`cu::Function::setCacheConfig` (C++ function), 27

`cu::HostMemory` (C++ class), 27

`cu::HostMemory::HostMemory` (C++ function), 27

`cu::HostMemory::operator T*` (C++ function), 27

`cu::HostMemory::size` (C++ function), 27

`cu::init` (C++ function), 36

`cu::memcpyDtoH` (C++ function), 36

`cu::memcpyHtoD` (C++ function), 36

`cu::Module` (C++ class), 28

`cu::Module::getGlobal` (C++ function), 28

`cu::Module::Module` (C++ function), 28

`cu::Module::optionmap_t` (C++ type), 28

`cu::Stream` (C++ class), 28

`cu::Stream::addCallback` (C++ function), 29

`cu::Stream::batchMemOp` (C++ function), 29

`cu::Stream::getContext` (C++ function), 29

`cu::Stream::launchKernel` (C++ function), 29

`cu::Stream::memAllocAsync` (C++ function), 29

`cu::Stream::memcpyDtoDAsync` (C++ function), 29

`cu::Stream::memcpyDtoHAsync` (C++ function), 29

`cu::Stream::memcpyHtoDAsync` (C++ function), 29

`cu::Stream::memcpyHtoHAsync` (C++ function), 29

cu::Stream::memFreeAsync (C++ *function*), 29
 cu::Stream::memPrefetchAsync (C++ *function*), 29
 cu::Stream::query (C++ *function*), 29
 cu::Stream::record (C++ *function*), 29
 cu::Stream::Stream (C++ *function*), 29
 cu::Stream::synchronize (C++ *function*), 29
 cu::Stream::wait (C++ *function*), 29
 cu::Stream::waitValue32 (C++ *function*), 29
 cu::Stream::writeValue32 (C++ *function*), 29
 cu::Stream::zero (C++ *function*), 29
 cu::Wrapper (C++ *class*), 30
 cu::Wrapper::_obj (C++ *member*), 30
 cu::Wrapper::manager (C++ *member*), 30
 cu::Wrapper::operator T (C++ *function*), 30
 cu::Wrapper::operator!= (C++ *function*), 30
 cu::Wrapper::operator== (C++ *function*), 30
 cu::Wrapper::Wrapper (C++ *function*), 30
 cufft::Error (C++ *class*), 31
 cufft::Error::Error (C++ *function*), 31
 cufft::Error::operator cufftResult (C++ *function*), 31
 cufft::Error::what (C++ *function*), 31
 cufft::FFT (C++ *class*), 31
 cufft::FFT1D (C++ *class*), 32
 cufft::FFT1D::FFT1D (C++ *function*), 32
 cufft::FFT2D (C++ *class*), 33
 cufft::FFT2D::FFT2D (C++ *function*), 33
 cufft::FFT::~~FFT (C++ *function*), 31
 cufft::FFT::checkCuFFTCall (C++ *function*), 32
 cufft::FFT::execute (C++ *function*), 31
 cufft::FFT::FFT (C++ *function*), 31
 cufft::FFT::operator= (C++ *function*), 31
 cufft::FFT::plan (C++ *function*), 32
 cufft::FFT::setStream (C++ *function*), 31
 nvtx::Marker::Color::yellow (C++ *enumerator*), 34
 nvtx::Marker::end (C++ *function*), 35
 nvtx::Marker::Marker (C++ *function*), 35
 nvtx::Marker::start (C++ *function*), 35

N

nvrtc::checkNvrtcCall (C++ *function*), 36
 nvrtc::Error (C++ *class*), 33
 nvrtc::Error::Error (C++ *function*), 33
 nvrtc::Error::operator nvrtcResult (C++ *function*), 33
 nvrtc::Error::what (C++ *function*), 33
 nvrtc::Program (C++ *class*), 34
 nvrtc::Program::~~Program (C++ *function*), 34
 nvrtc::Program::compile (C++ *function*), 34
 nvrtc::Program::getLog (C++ *function*), 34
 nvrtc::Program::getPTX (C++ *function*), 34
 nvrtc::Program::Program (C++ *function*), 34
 nvtx::Marker (C++ *class*), 34
 nvtx::Marker::Color (C++ *enum*), 34
 nvtx::Marker::Color::black (C++ *enumerator*), 34
 nvtx::Marker::Color::blue (C++ *enumerator*), 34
 nvtx::Marker::Color::green (C++ *enumerator*), 34
 nvtx::Marker::Color::red (C++ *enumerator*), 34