Dynamic Kernel Scheduler

Version 1

Generated by Doxygen 1.6.1

Thu Mar 5 23:56:21 2015

Contents

Chapter 1

Class 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

Class Documentation

3.1 Device Class Reference

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

- /home/l_locans/svnwork/phd/DKS/src/DKSDevice.h

3.2 DKSBase Class Reference

DKSBase class for handling function calls to DKS library.

`#include <DKSBase.h>`

Public Member Functions

- DKSBase ()

  Default constructor.
- DKSBase (const char ∗api_name, const char ∗device_name)

  Constructor that sets api and devcie to use with DKS.
- ∼DKSBase ()

  Destructor.
- int setDevice (const char ∗device_name, int length)

  Set device to use with DKS.
- int setAPI (const char ∗api_name, int length)

  Set framework to use with DKS.
- int getDevices ()

  Prints information about all available devices.
- int getDeviceCount ()

  Returns device count.
- int setFunction (const char ∗function_name, int length)

  Set OpenCL function to use, loads necessary OpenCL kernel file.
- int initDevice ()

  Inititialize DKS.
- int createStream (int &streamId)

  Create stream for async execution.
- int closeHandle (void ∗mem_ptr)

  Send pointer to device memory from one MPI process to another.
- int syncDevice ()

  Wait till all tasks running on device are completed.
- template<typename T >
  void ∗ pushData (const void ∗data_in, int elements, int &ierr)

  Allocate memory and transfer data to device.
- template<typename T >
  int pullData (void ∗mem_ptr, void ∗data_out, int elements)

Read data from device and free device memory.

- template<typename T >
  void ∗ allocateMemory (int elements, int &ierr)

  Allocate memory on device and return pointer to device memory.
- template<typename T >
  int allocateHostMemory (T ∗&ptr, int size)

  Allocates host memory as page-locked.
- template<typename T >
  int freeHostMemory (T ∗&ptr, int size)

  Free host page-locked memory.
- template<typename T >
  int writeData (void ∗mem_ptr, const void ∗data, int elements, int offset=0)

  Write data from host to device.
- template<typename T >
  int writeDataAsync (void ∗mem_ptr, const void ∗data, int elements, int streamId=-1, int offset=0)

  Write data to device using async write.
- template<typename T >
  int readData (const void ∗mem_ptr, void ∗out_data, int elements, int offset=0)

  Gather 3D data from multiple mpi processes to one memory region.
- template<typename T >
  int readDataAsync (const void ∗mem_ptr, void ∗out_data, int elements, int streamId=-1, int offset=0)

  Performs an async data read from device.
- template<typename T >
  int freeMemory (void ∗mem_ptr, int elements)

  Free memory allocated on device.
- int setupFFT (int ndim, int N[3])

  Setup FFT function.
- int callFFT (void ∗data_ptr, int ndim, int dimsize[3], int streamId=-1)

  Call complex-to-complex fft.
- int callIFFT (void ∗data_ptr, int ndim, int dimsize[3], int streamId=-1)

  Call complex-to-complex ifft.
- int callNormalizeFFT (void ∗data_ptr, int ndim, int dimsize[3], int streamId=-1)

  Normalize complex to complex ifft.
- int callFFTStockham (void ∗&src, int ndim, int dimsize[3])

  Test function for stockham FFT implementation using OpenCL.
- int callR2CFFT (void ∗real_ptr, void ∗comp_ptr, int ndim, int dimsize[3], int streamId=-1)

Call real to complex FFT.

- int callC2RFFT (void ∗real_ptr, void ∗comp_ptr, int ndim, int dimsize[3], int streamId=-1)

  Call complex to real iFFT.

- int callNormalizeC2RFFT (void ∗real_ptr, int ndim, int dimsize[3], int streamId=-1)

  Normalize compelx to real ifft.

- int callGreensIntegral (void ∗tmp_ptr, int I, int J, int K, int NI, int NJ, double hz_m0, double hz_m1, double hz_m2, int streamId=-1)

  Integrated greens function from OPAL FFTPoissonsolver.cpp put on device.

- int callGreensIntegration (void ∗mem_ptr, void ∗tmp_ptr, int I, int J, int K, int streamId=-1)

  Integrated greens function from OPAL FFTPoissonsolver.cpp put on device.

- int callMirrorRhoField (void ∗mem_ptr, int I, int J, int K, int streamId=-1)

  Integrated greens function from OPAL FFTPoissonsolver.cpp put on device.

- int callMultiplyComplexFields (void ∗mem_ptr1, void ∗mem_ptr2, int size, int streamId=-1)

  Element by element multiplication.

- int callPHistoTFFcn (void ∗mem_data, void ∗mem_par, void ∗mem_chisq, double fTimeResolution, double fRebin, int sensors, int length, int numpar, double &result)

  Chi square for parameter fitting on device.

- int callSingleGaussTF (void ∗mem_data, void ∗mem_t0, void ∗mem_par, void ∗mem_result, double fTimeResolution, double fRebin, double fGoodBinOffser, int sensors, int length, int numpar, double &result)

  max-log-likelihood for parameter fitting on device.

- int callDoubleLorentzTF (void ∗mem_data, void ∗mem_t0, void ∗mem_par, void ∗mem_result, double fTimeResolution, double fRebin, double fGoodBinOffser, int sensors, int length, int numpar, double &result)

  max-log-likelihood for parameter fitting on device.

- int callCollimatorPhysics (void ∗mem_ptr, void ∗par_ptr, int numparticles, int numparams, int &numaddback, int &numdead)

  Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device.

- int callInitRandoms (int size)

  Init random number states and save for reuse on device.

- void oclEventInfo ()

  Test function to profile opencl kernel calls.

- void oclClearEvents ()

  Test function to profile opencl kernel calls.

3.2.1 Detailed Description

DKSBase class for handling function calls to DKS library.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 DKSBase::~DKSBase ()

Destructor. Free DKS resources.

3.2.3 Member Function Documentation

3.2.3.1 template<typename T > int DKSBase::allocateHostMemory (T ∗& *ptr*, int *size*) `[inline]`

Allocates host memory as page-locked. Used for memroy allocation on the host side for pointer ptr for size elements. Page locked memory improves data transfer rates between host and device and allows async data transfer and kernel execution. Reurns succes or error code. TODO: opencl and mic implementations needed.

3.2.3.2 template<typename T > void∗ DKSBase::allocateMemory (int *elements*, int & *ierr*) `[inline]`

Allocate memory on device and return pointer to device memory. Allocates memory of type T, elements specifies the number of elements for which memory should be allocated. If memory allocation fails ierr is set to error code. Returns void pointer to device memory.

3.2.3.3 int DKSBase::callC2RFFT (void ∗ *real_ptr*, void ∗ *comp_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Call complex to real iFFT. Executes out of place complex to real ifft, real_ptr points to real data, comp_pt - points to complex data, ndim - dimension of data, dimsize size of each dimension. real_ptr size should be dimsize[0]∗dimsize[1]∗disize[2], comp_ptr size should be atleast (dimsize[0]/2+1)∗dimsize[1]∗dimsize[2] TODO: opencl and mic implementations.

3.2.3.4 int DKSBase::callCollimatorPhysics (void ∗ *mem_ptr*, void ∗ *par_ptr*, int *numparticles*, int *numparams*, int & *numaddback*, int & *numdead*)

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs. TODO: opencl and mic implementations.

3.2.3.5 int DKSBase::callDoubleLorentzTF (void ∗ *mem_data*, void ∗ *mem_t0*, void ∗ *mem_par*, void ∗ *mem_result*, double *fTimeResolution*, double *fRebin*, double *fGoodBinOffser*, int *sensors*, int *length*, int *numpar*, double & *result*)

max-log-likelihood for parameter fitting on device. mem_data - measurement data, mem_t0 - pointer to time 0 for each sensor, mem_par - pointer to parameter set, mem_results - pointer for intermediate results. Chi square results are put in &results. TODO: opencl and mic implementations.

3.2.3.6 int DKSBase::callFFT (void ∗ *data_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Call complex-to-complex fft. Executes in place complex to compelx fft on the device on data pointed by data_ptr. stream id can be specified to use other streams than default. TODO: mic implementation

3.2.3.7 int DKSBase::callFFTStockham (void ∗& *src*, int *ndim*, int *dimsize*[3])

Test function for stockham FFT implementation using OpenCL. Not finished and tested - should no be used.

3.2.3.8 int DKSBase::callGreensIntegral (void ∗ *tmp_ptr*, int *I*, int *J*, int *K*, int *NI*, int *NJ*, double *hz_m0*, double *hz_m1*, double *hz_m2*, int *streamId* = `-1`)

Integrated greens function from OPAL FFTPoissonsolver.cpp put on device. For specifics check OPAL docs. TODO: opencl and mic implementations.

3.2.3.9 int DKSBase::callGreensIntegration (void ∗ *mem_ptr*, void ∗ *tmp_ptr*, int *I*, int *J*, int *K*, int *streamId* = `-1`)

Integrated greens function from OPAL FFTPoissonsolver.cpp put on device. For specifics check OPAL docs. TODO: opencl and mic implementations.

3.2.3.10 int DKSBase::callIFFT (void ∗ *data_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Call complex-to-complex ifft. Executes in place complex to compelx ifft on the device on data pointed by data_ptr. stream id can be specified to use other streams than default. TODO: mic implementation.

3.2.3.11 int DKSBase::callInitRandoms (int *size*)

Init random number states and save for reuse on device. TODO: opencl and mic implementations.

3.2.3.12 int DKSBase::callMirrorRhoField (void ∗ *mem_ptr*, int *I*, int *J*, int *K*, int *streamId* = `-1`)

Integrated greens function from OPAL FFTPoissonsolver.cpp put on device. For specifics check OPAL docs. TODO: opencl and mic implementations.

3.2.3.13 int DKSBase::callMultiplyComplexFields (void ∗ *mem_ptr1*, void ∗ *mem_ptr2*, int *size*, int *streamId* = `-1`)

Element by element multiplication. Multiplies each element of mem_ptr1 with corresponding element of mem_ptr2, size specifies the number of elements in mem_ptr1 and mem_ptr2 to use. Results are put in mem_ptr1. TODO: opencl and mic implementations.

3.2.3.14 int DKSBase::callNormalizeC2RFFT (void ∗ *real_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Normalize compelx to real ifft. Cuda, mic and OpenCL implementations return ifft unscaled, this function divides each element by fft size. TODO: opencl and mic implementations.

3.2.3.15 int DKSBase::callNormalizeFFT (void ∗ *data_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Normalize complex to complex ifft. Cuda, mic and OpenCL implementations return ifft unscaled, this function divides each element by fft size TODO: mic implementation.

3.2.3.16 int DKSBase::callPHistoTFFcn (void ∗ *mem_data*, void ∗ *mem_par*, void ∗ *mem_chisq*, double *fTimeResolution*, double *fRebin*, int *sensors*, int *length*, int *numpar*, double & *result*)

Chi square for parameter fitting on device. mem_data - measurement data, mem_par - pointer to parameter set, mem_chisq - pointer for intermediate results. Chi square results are put in &results

3.2.3.17 int DKSBase::callR2CFFT (void ∗ *real_ptr*, void ∗ *comp_ptr*, int *ndim*, int *dimsize*[3], int *streamId* = `-1`)

Call real to complex FFT. Executes out of place real to complex fft, real_ptr points to real data, comp_pt - points to complex data, ndim - dimension of data, dimsize size of each dimension. real_ptr size should be dimsize[0]∗dimsize[1]∗disize[2], comp_ptr size should be atleast (dimsize[0]/2+1)∗dimsize[1]∗dimsize[2] TODO: opencl and mic implementations

3.2.3.18 int DKSBase::callSingleGaussTF (void ∗ *mem_data*, void ∗ *mem_t0*, void ∗ *mem_par*, void ∗ *mem_result*, double *fTimeResolution*, double *fRebin*, double *fGoodBinOffser*, int *sensors*, int *length*, int *numpar*, double & *result*)

max-log-likelihood for parameter fitting on device. mem_data - measurement data, mem_t0 - pointer to time 0 for each sensor, mem_par - pointer to parameter set, mem_results - pointer for intermediate results. Chi square results are put in &results. TODO: opencl and mic implementations.

3.2.3.19 int DKSBase::closeHandle (void ∗ *mem_ptr*)

Send pointer to device memory from one MPI process to another. Implemented only if mpi compiler is used to build DKS. Implemented only for cuda. Uses cuda icp. Gets icp handle of memory allocated on device pointed by mem_ptr does MPI_Send to dest process where matching receivePointer should be called. Returns success or error code. TODO: opencl and mic cases still need implementations Receive pointer to device memory from another MPI process. Implemented only if mpi compiler is used to build DKS. Implemented only for cuda. Uses cuda icp. Uses MPI_Recv to get icp handle from another MPI process and opens a reference to this memory. Togeter with sendPointer function allows multiple MPI processes to share one memory region of the device. Returns success or error code. TODO: opencl and mic cases still need implementations Close handle to device memory. If receivePointer is used to open memory handle allocated by another MPI process closeHandle should be called to free resources instead of freeMemory. Returns success or error code. TODO: opencl and mic cases still need implementations.

3.2.3.20 int DKSBase::createStream (int & *streamId*)

Create stream for async execution. Function to create different streams with device to allow assync kernel execution and data transfer. Currently implemented for CUDA with cuda streams. streamId will be can be used later use the created stream. Returns success or error code. TODO: for opencl use different contexts similar as cuda streams to achieve async execution. TODO: for intel mic look at library (libxstream) from Hans Pabst.

3.2.3.21 template<typename T > int DKSBase::freeHostMemory (T ∗& *ptr*, int *size*) `[inline]`

Free host page-locked memory. Used to free page-locked memory on the host that was allocated using allocateHostMemory. ptr is the host pointer where page-locked memory was allocated, size - number of elements held by the memroy.

3.2.3.22 template<typename T > int DKSBase::freeMemory (void ∗ *mem_ptr*, int *elements*) `[inline]`

Free memory allocated on device. Free memory referenced by mem_ptr, elements - number of elements in memory, T - data type.

3.2.3.23 int DKSBase::getDeviceCount () `[inline]`

Returns device count. Not implemented jet, returns 0

3.2.3.24 int DKSBase::getDevices ()

Prints information about all available devices. Calls CUDA, OpenCL and MIC functions to query for available devices for each framework and pirnts information about each device. Returns success or error code

3.2.3.25 int DKSBase::initDevice ()

Inititialize DKS. Set framework and device to use. If OpenCL is used create context with device. Return success or error code.

3.2.3.26 void DKSBase::oclClearEvents () `[inline]`

Test function to profile opencl kernel calls. Used for debuging and timing purposes only.

3.2.3.27 void DKSBase::oclEventInfo () `[inline]`

Test function to profile opencl kernel calls. Used for debuging and timing purposes only.

3.2.3.28 template<typename T > int DKSBase::pullData (void ∗ *mem_ptr*, void ∗ *data_out*, int *elements*) `[inline]`

Read data from device and free device memory. Reads data from device pointed by mem_ptr into data_out pointer. Elements specifies the number of data elements to read, T specifies the datatype of elements to copy. Returns error code if read data or free memory fails.

3.2.3.29 template<typename T > void∗ DKSBase::pushData (const void ∗ *data_in*, int *elements*, int & *ierr*) `[inline]`

Allocate memory and transfer data to device. Returns a void pointer which can be used in later kernels to reference allocated device memory. data_in pointer to data to be transfered to device, elements is the number of data elements to transfer, T - type of data to transfer. If memory allocation or data transfer fails ierr will be set to error code.

3.2.3.30 template<typename T > int DKSBase::readData (const void ∗ *mem_ptr*, void ∗ *out_data*, int *elements*, int *offset* = `0`) `[inline]`

Gather 3D data from multiple mpi processes to one memory region. When multiple processes share the same device memory using sendPointer and receivePointer gather3DDataAsync allows each process to write data to its memory region. Uses async writes. mem_ptr - device pointer, data - host pointer, Ng - global dimensions of data, Nl - local data dimensions, id - starting indexes in global domain for each process streamId - stream to use for data transfers. Returns success or error code. Scatter 3D data to multiple MPI processes from one device memory region. When multiple processes share the same device memory using sendPointer and receivePointer scatter3DDataAsync allows each process to read data from its memory region. Uses async reads. mem_ptr - device pointer, data - host pointer, Ng - global dimensions of data, Nl - local data dimensions, id - starting indexes in global domain for each process streamId - stream to use for data transfers. Returns success or error code. Create MPI subarray for 3D data gather and scatter using cuda aware MPI. If multiple MPI processes share device and cuda aware MPI is used for data transfer creates a MPI subarray so each MPI process can write and read to its own memory region. N_global - global domain dimensions, N_local - local domain dimensions, datatype - MPI datatype Gather 3D data from multiple MPI processes to device using cuda aware MPI. Using cuda aware mpi allows to gather data to one device memory region allocated by one of the mpi processes. mem_ptr - device pointer, data - host memory pointer, size - number of elements to transfer, stype - data type of elements, N_global - global dimensions of the domain, N_local - local domain dimensions, idx,idy,idz - starting indexes in global domain for each process, numNodes - number of processes, myNode - current node, rootNode - node that allocated device memory, comm - MPI communicator TODO: opencl and mic implementations (solution other than cuda aware mpi needed). Gather 3D data from multiple MPI processes to device using cuda aware MPI and non blocking gather. For detailed parameter description see gather3DData docs. TODO: opencl and mic implementations (solution other than cuda aware mpi needed). Scatter 3D data from device to multiple MPI processes using cuda aware MPI. If multiple MPI prcesses share one device allows to scatter 3D data regions from device memory allocated by one of the processes to all other MPI processes. For detailed parameter description see gather3DData docs. TODO: opencl and mic implementations (solution other than cuda aware mpi needed). Read data from device memory. Read data referenced by mem_ptr int out_data. Elements indicates the number of data elements to read and offset is the offset on the device from start of the memroy. Data type to read is specified by T. Performs a blocking read.

3.2.3.31 template<typename T > int DKSBase::readDataAsync (const void ∗ *mem_ptr*, void ∗ *out_data*, int *elements*, int *streamId* = `-1`, int *offset* = `0`) `[inline]`

Performs an async data read from device. Queues data read from device and returns control to host. stream id specifies stream to use for the read. Device async read can be performed if host memroy is page-locked and strema other than default -1 is used. For other parameter detailed description see readData function. TODO: opencl and mic implementations (currently reverts to blocking reads).

3.2.3.32 int DKSBase::setAPI (const char ∗ *api_name*, int *length*)

Set framework to use with DKS. Sets framework and API that DKS uses to execute code on device. Supported API's are OpenCL, CUDA and OpenMP. Returns success or error code

3.2.3.33 int DKSBase::setDevice (const char ∗ *device_name*, int *length*)

Set device to use with DKS. Sets specific device to use with DKS. Supported devices are -gpu and -mic. Return success or error code.

3.2.3.34 int DKSBase::setFunction (const char ∗ *function_name*, int *length*)

Set OpenCL function to use, loads necessary OpenCL kernel file. Deprecated function, not really used anymore

3.2.3.35 int DKSBase::setupFFT (int *ndim*, int *N*[3])

Setup FFT function. Initializes parameters for fft executuin. If ndim > 0 initializes handles for fft calls. If ffts of various sizes are needed setupFFT should be called with ndim 0, in this case each fft will do its own setup according to fft size and dimensions. TODO: opencl and mic implementations

3.2.3.36 int DKSBase::syncDevice ()

Wait till all tasks running on device are completed. Forces a device synchronization - waits till all tasks on the device are complete. Implemented for cuda. Forces sync only in context in witch it is called - only waits for tasks launched by process calling syncDevice. If multiple processes launch different tasks each process is responsible for its own synchronization. Returns success or error code. TODO: opencl and mic implementations still necessary

3.2.3.37 template<typename T > int DKSBase::writeData (void ∗ *mem_ptr*, const void ∗ *data*, int *elements*, int *offset* = `0`) `[inline]`

Write data from host to device. Write data from data to device memory referenced by mem_ptr. Elements spicify the number of elements to write, offset specifies the offset from the first element. Returns success or error code. Performs a blocking write - control to the host is returned only when data transfer is complete.

3.2.3.38 template<typename T > int DKSBase::writeDataAsync (void ∗ *mem_ptr*, const void ∗ *data*, int *elements*, int *streamId* = `-1`, int *offset* = `0`) `[inline]`

Write data to device using async write. Queue a async data write and return control to host imediately. mem_ptr - device memory pointer, data - host memory pointer, elements - number of data elements to write stremaId - stream id to use, offset - offset on device from first element For trully async execution on cuda stream other than default needs to be created and device memory must be page-locked. Otherwise functions just asynchronosly with respect to host. TODO: mic and opencl implementations needed (goes to blocking writes)

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

- /home/l_locans/svnwork/phd/DKS/src/DKSBase.h
- /home/l_locans/svnwork/phd/DKS/src/DKSBase.cpp

3.3 DKSStream Class Reference

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

- /home/l_locans/svnwork/phd/DKS/src/DKSStream.h

3.4 MICBase Class Reference

Inherited by MICChiSquare.

Public Member Functions

- int **mic_setDeviceId** (int id)
- int **mic_getDevices** ()
- template<typename T >
  void $*$ **mic_allocateMemory** (int size)
- template<typename T >
  int **mic_writeData** (void $*$data_ptr, const void $*$data, int size, int offset=0)
- template<typename T >
  int **mic_readData** (const void $*$data_ptr, void $*$result, int size, int offset=0)
- template<typename T >
  int **mic_freeMemory** (void $*$data_ptr, int size)
- template<typename T >
  void $*$ **mic_pushData** (const void $*$data, int size)
- template<typename T >
  int **mic_pullData** (void $*$data_ptr, void $*$result, int size)

Protected Attributes

- int **m_device_id**

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

- /home/l_locans/svnwork/phd/DKS/src/MIC/MICBase.h
- /home/l_locans/svnwork/phd/DKS/src/MIC/MICBase.cpp

3.5 MICChiSquare Class Reference

Inherits MICBase.

Public Member Functions

- int **mic_chi2** (double ∗O, double ∗E, double ∗result, int size)
- int **mic_Nt** (double ∗nt, double ∗p, int psize, int nsize, int jsize, double deltaT=1)
- int **mic_sum** (double ∗data, double ∗result, int size)

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

- /home/l_locans/svnwork/phd/DKS/src/MIC/MICChiSquare.h
- /home/l_locans/svnwork/phd/DKS/src/MIC/MICChiSquare.cpp

3.6 OpenCLBase Class Reference

Inherited by OpenCLChiSquare, and OpenCLFFT.

Public Member Functions

- int **ocl_getAllDevices** ()
- int **ocl_setUp** (const char ∗device_name)
- int **ocl_loadKernel** (const char ∗kernel_file)
- cl_mem **ocl_allocateMemory** (size_t size, int &ierr)
- cl_mem **ocl_allocateMemory** (size_t size, int type, int &ierr)
- int **ocl_writeData** (cl_mem mem_ptr, const void ∗in_data, size_t size, size_t offset=0, int blocking=CL_TRUE)
- int **ocl_copyData** (cl_mem src_ptr, cl_mem dst_ptr, size_t size)
- int **ocl_createKernel** (const char ∗kernel_name)
- int **ocl_setKernelArg** (int idx, size_t size, const void ∗arg_value)
- int **ocl_executeKernel** (cl_uint, const size_t ∗work_items, const size_t ∗work_grou_size=NULL)
- int **ocl_readData** (cl_mem mem_ptr, void ∗out_data, size_t size, size_t offset=0, int blocking=CL_-TRUE)
- int **ocl_freeMemory** (cl_mem mem_ptr)
- int **ocl_cleanUp** ()
- int **ocl_deviceInfo** (bool verbose=true)
- void **ocl_clearEvents** ()
- void **ocl_eventInfo** ()

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

- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLBase.h
- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLBase.cpp

3.7 OpenCLChiSquare Class Reference

Inherits OpenCLBase.

Public Member Functions

- int **ocl_PHistoTFFcn** (void ∗mem_data, void ∗mem_par, void ∗mem_result, double fTimeResolution, double fRebin, int sensors, int length, int numpar, double &result)

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

- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLChiSquare.h
- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLChiSquare.cpp

3.8 OpenCLFFT Class Reference

Inherits OpenCLBase.

Public Member Functions

- int **ocl_executeFFT** (void ∗data, int ndim, int N, bool forward=true)
- int **ocl_executeIFFT** (void ∗data, int ndim, int N)
- int **ocl_normalizeFFT** (void ∗data, int ndim, int N)
- int **ocl_setupFFT** ()
- int **ocl_executeFFTStockham** (void ∗&src, int ndim, int N, bool forward=true)
- int **ocl_executeFFTStockham2** (void ∗&src, int ndim, int N, bool forward=true)
- int **ocl_executeTranspose** (cl_mem src, int N, int ndim, int dim)
- void **printData3DN4** (cl_double2 ∗&data, int N)

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

- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLFFT.h
- /home/l_locans/svnwork/phd/DKS/src/OpenCL/OpenCLFFT.cpp

Index