

Dynamic Kernel Scheduler

DKS 1.1.1

Generated by Doxygen 1.8.6

Mon May 29 2017 13:19:15

Contents

1	Hierarchical Index	1
1.1	Class Hierarchy	1
2	Class Index	3
2.1	Class List	3
3	Class Documentation	5
3.1	BaseFFT Class Reference	5
3.2	ChiSquareRuntime Class Reference	6
3.2.1	Constructor & Destructor Documentation	7
3.2.1.1	~ChiSquareRuntime	7
3.2.2	Member Function Documentation	7
3.2.2.1	getOperations	7
3.2.2.2	setConsts	7
3.2.2.3	setConsts	7
3.2.2.4	setKernelParams	7
3.3	compare_particle Struct Reference	7
3.4	compare_particle_small Struct Reference	8
3.5	CUDA_PART2 Struct Reference	8
3.5.1	Detailed Description	8
3.6	CUDA_PART2_SMALL Struct Reference	8
3.6.1	Detailed Description	9
3.7	CUDA_PART_SMALL Struct Reference	9
3.7.1	Detailed Description	9
3.8	CudaBase Class Reference	9
3.8.1	Member Function Documentation	10
3.8.1.1	cuda_addStream	10
3.8.1.2	cuda_allocateHostMemory	10
3.8.1.3	cuda_allocateMemory	10
3.8.1.4	cuda_cleanUp	10
3.8.1.5	cuda_createCurandStates	11
3.8.1.6	cuda_createRandomNumbers	11

3.8.1.7	cuda_createStream	11
3.8.1.8	cuda_defaultStream	11
3.8.1.9	cuda_deleteCurandStates	11
3.8.1.10	cuda_deleteStream	11
3.8.1.11	cuda_deleteStreams	11
3.8.1.12	cuda_executeFunction	11
3.8.1.13	cuda_freeHostMemory	11
3.8.1.14	cuda_freeMemory	11
3.8.1.15	cuda_getCublas	11
3.8.1.16	cuda_getCurandStates	11
3.8.1.17	cuda_getDeviceCount	12
3.8.1.18	cuda_getDeviceName	12
3.8.1.19	cuda_getDevices	12
3.8.1.20	cuda_getStream	12
3.8.1.21	cuda_getStreamId	12
3.8.1.22	cuda_getUniqueDevices	12
3.8.1.23	cuda_hostRegister	12
3.8.1.24	cuda_hostUnregister	12
3.8.1.25	cuda_memInfo	12
3.8.1.26	cuda_numberOfStreams	12
3.8.1.27	cuda_pullData	12
3.8.1.28	cuda_pushData	12
3.8.1.29	cuda_readData	13
3.8.1.30	cuda_readDataAsync	13
3.8.1.31	cuda_setDevice	13
3.8.1.32	cuda_setStream	13
3.8.1.33	cuda_setUp	13
3.8.1.34	cuda_syncDevice	13
3.8.1.35	cuda_writeData	13
3.8.1.36	cuda_writeDataAsync	13
3.8.1.37	cuda_zeroMemory	13
3.8.1.38	cuda_zeroMemoryAsync	13
3.9	CudaChiSquare Class Reference	14
3.9.1	Constructor & Destructor Documentation	14
3.9.1.1	CudaChiSquare	14
3.10	CudaChiSquareRuntime Class Reference	14
3.10.1	Constructor & Destructor Documentation	15
3.10.1.1	CudaChiSquareRuntime	15
3.10.1.2	CudaChiSquareRuntime	15
3.10.1.3	~CudaChiSquareRuntime	15

3.10.2	Member Function Documentation	15
3.10.2.1	checkChiSquareKernels	15
3.10.2.2	compileProgram	15
3.10.2.3	freeChiSquare	15
3.10.2.4	initChiSquare	15
3.10.2.5	launchChiSquare	15
3.10.2.6	writeFunc	16
3.10.2.7	writeMap	16
3.10.2.8	writeParams	16
3.11	CudaCollimatorPhysics Class Reference	16
3.11.1	Detailed Description	17
3.11.2	Constructor & Destructor Documentation	17
3.11.2.1	CudaCollimatorPhysics	17
3.11.2.2	CudaCollimatorPhysics	17
3.11.2.3	~CudaCollimatorPhysics	17
3.11.3	Member Function Documentation	17
3.11.3.1	CollimatorPhysics	17
3.11.3.2	CollimatorPhysicsSort	17
3.11.3.3	ParallelTrackerPush	17
3.11.3.4	ParallelTrackerPushTransform	17
3.12	CudaFFT Class Reference	18
3.12.1	Constructor & Destructor Documentation	18
3.12.1.1	CudaFFT	18
3.12.1.2	CudaFFT	18
3.12.1.3	~CudaFFT	18
3.12.2	Member Function Documentation	19
3.12.2.1	destroyFFT	19
3.12.2.2	setupFFT	19
3.13	CudaGreensFunction Class Reference	19
3.13.1	Constructor & Destructor Documentation	19
3.13.1.1	CudaGreensFunction	19
3.13.2	Member Function Documentation	19
3.13.2.1	greensIntegral	19
3.13.2.2	integrationGreensFunction	20
3.13.2.3	mirrorRhoField	20
3.13.2.4	multiplyComplexFields	20
3.14	CudaImageReconstruction Class Reference	20
3.14.1	Constructor & Destructor Documentation	21
3.14.1.1	CudaImageReconstruction	21
3.14.1.2	CudaImageReconstruction	21

3.14.1.3	~CudalImageReconstruction	21
3.14.2	Member Function Documentation	21
3.14.2.1	backwardProjection	21
3.14.2.2	calculateBackground	21
3.14.2.3	calculateBackgrounds	21
3.14.2.4	calculateSource	21
3.14.2.5	calculateSources	22
3.14.2.6	forwardProjection	22
3.14.2.7	generateNormalization	22
3.14.2.8	setDimensions	22
3.14.2.9	setEdge	22
3.14.2.10	setEdge1	22
3.14.2.11	setMinCrystallnRing	22
3.14.2.12	setParams	22
3.15	Device Class Reference	23
3.16	DKSAutoTuning Class Reference	23
3.16.1	Constructor & Destructor Documentation	23
3.16.1.1	DKSAutoTuning	23
3.16.1.2	~DKSAutoTuning	23
3.16.2	Member Function Documentation	23
3.16.2.1	addParameter	23
3.16.2.2	clearParameters	23
3.16.2.3	exhaustiveSearch	23
3.16.2.4	hillClimbing	24
3.16.2.5	lineSearch	24
3.16.2.6	setFunction	24
3.16.2.7	simulatedAnnealing	24
3.17	DKSAutoTuningTester Class Reference	24
3.18	DKSBase Class Reference	24
3.18.1	Detailed Description	26
3.18.2	Constructor & Destructor Documentation	26
3.18.2.1	DKSBase	26
3.18.2.2	DKSBase	26
3.18.2.3	~DKSBase	26
3.18.3	Member Function Documentation	26
3.18.3.1	allocateHostMemory	26
3.18.3.2	allocateMemory	26
3.18.3.3	apiCuda	26
3.18.3.4	apiOpenCL	27
3.18.3.5	apiOpenMP	27

3.18.3.6	callCreateRandomNumbers	27
3.18.3.7	callInitRandoms	27
3.18.3.8	callMemInfo	27
3.18.3.9	closeHandle	27
3.18.3.10	createStream	27
3.18.3.11	deviceCPU	27
3.18.3.12	deviceGPU	27
3.18.3.13	deviceMIC	27
3.18.3.14	freeHostMemory	28
3.18.3.15	freeMemory	28
3.18.3.16	getDeviceCount	28
3.18.3.17	getDeviceList	28
3.18.3.18	getDeviceName	28
3.18.3.19	getDevices	28
3.18.3.20	initDevice	28
3.18.3.21	isAutoTuningOn	28
3.18.3.22	isUseConfigOn	28
3.18.3.23	loadOpenCLKernel	28
3.18.3.24	oclClearEvents	29
3.18.3.25	oclEventInfo	29
3.18.3.26	pullData	29
3.18.3.27	pushData	29
3.18.3.28	readData	29
3.18.3.29	readDataAsync	29
3.18.3.30	registerHostMemory	30
3.18.3.31	setAPI	30
3.18.3.32	setAutoTuningOff	30
3.18.3.33	setAutoTuningOn	30
3.18.3.34	setDefaultDevice	30
3.18.3.35	setDevice	30
3.18.3.36	setupDevice	30
3.18.3.37	setUseConfigOff	30
3.18.3.38	setUseConfigOn	30
3.18.3.39	syncDevice	30
3.18.3.40	unregisterHostMemory	31
3.18.3.41	writeData	31
3.18.3.42	writeDataAsync	31
3.19	DKSBaseMuSR Class Reference	31
3.19.1	Member Function Documentation	32
3.19.1.1	callAutoTuningChiSquare	32

3.19.1.2	callCompileProgram	32
3.19.1.3	callLaunchChiSquare	32
3.19.1.4	callSetConsts	32
3.19.1.5	callSetConsts	32
3.19.1.6	checkMuSRKernels	32
3.19.1.7	checkMuSRKernels	33
3.19.1.8	freeChiSquare	33
3.19.1.9	getOperations	33
3.19.1.10	initChiSquare	33
3.19.1.11	testAutoTuning	33
3.19.1.12	writeFunctions	33
3.19.1.13	writeMaps	33
3.19.1.14	writeParams	33
3.20	DKSCollimatorPhysics Class Reference	33
3.21	DKSConfig Class Reference	34
3.21.1	Constructor & Destructor Documentation	34
3.21.1.1	DKSConfig	34
3.21.2	Member Function Documentation	34
3.21.2.1	addConfigParameter	34
3.21.2.2	getConfigParameter	35
3.21.2.3	loadConfigFile	35
3.21.2.4	saveConfigFile	35
3.22	DKSFFT Class Reference	35
3.22.1	Member Function Documentation	35
3.22.1.1	callC2RFFT	35
3.22.1.2	callIFFT	36
3.22.1.3	callIIFFT	36
3.22.1.4	callNormalizeC2RFFT	36
3.22.1.5	callNormalizeFFT	36
3.22.1.6	callR2CFFT	36
3.22.1.7	setupFFT	36
3.23	DKSImageRecon Class Reference	36
3.23.1	Member Function Documentation	37
3.23.1.1	callBackwardProjection	37
3.23.1.2	callCalculateBackground	37
3.23.1.3	callCalculateBackgrounds	37
3.23.1.4	callCalculateSource	37
3.23.1.5	callCalculateSources	37
3.23.1.6	callForwardProjection	37
3.23.1.7	callGenerateNormalization	38

3.23.1.8	setDimensions	38
3.23.1.9	setEdge	38
3.23.1.10	setEdge1	38
3.23.1.11	setMinCrystallnRing	38
3.23.1.12	setParams	38
3.24	DKSOPAL Class Reference	38
3.24.1	Member Function Documentation	39
3.24.1.1	callCollimatorPhysics	39
3.24.1.2	callCollimatorPhysics2	39
3.24.1.3	callCollimatorPhysicsSoA	39
3.24.1.4	callCollimatorPhysicsSort	39
3.24.1.5	callCollimatorPhysicsSortSoA	40
3.24.1.6	callGreensIntegral	40
3.24.1.7	callGreensIntegration	40
3.24.1.8	callMirrorRhoField	40
3.24.1.9	callMultiplyComplexFields	40
3.24.1.10	callParallelTTrackerKick	40
3.24.1.11	callParallelTTrackerPush	40
3.24.1.12	callParallelTTrackerPush	40
3.24.1.13	callParallelTTrackerPushTransform	40
3.25	DKSSearchStates Class Reference	41
3.25.1	Constructor & Destructor Documentation	41
3.25.1.1	DKSSearchStates	41
3.25.2	Member Function Documentation	41
3.25.2.1	getCurrentState	41
3.25.2.2	getNeighbours	41
3.25.2.3	getNextNeighbour	41
3.25.2.4	getRandomNeighbour	41
3.25.2.5	initCurrentState	42
3.25.2.6	moveToNeighbour	42
3.25.2.7	nextNeighbourExists	42
3.25.2.8	printBest	42
3.25.2.9	printCurrentState	42
3.25.2.10	printInfo	42
3.25.2.11	printNeighbour	42
3.25.2.12	saveCurrentState	42
3.25.2.13	setCurrentState	42
3.25.2.14	setCurrentState	42
3.26	DKSStream Class Reference	43
3.27	DKSTimer Class Reference	43

3.27.1	Constructor & Destructor Documentation	43
3.27.1.1	DKSTimer	43
3.27.2	Member Function Documentation	43
3.27.2.1	gettime	43
3.27.2.2	init	43
3.27.2.3	print	43
3.27.2.4	reset	43
3.27.2.5	start	43
3.27.2.6	stop	44
3.28	Double3 Struct Reference	44
3.29	GreensFunction Class Reference	44
3.29.1	Member Function Documentation	44
3.29.1.1	greensIntegral	44
3.29.1.2	integrationGreensFunction	45
3.29.1.3	mirrorRhoField	45
3.29.1.4	multiplyCompelxFields	45
3.30	ImageReconstruction Class Reference	45
3.30.1	Member Function Documentation	46
3.30.1.1	backwardProjection	46
3.30.1.2	calculateBackground	46
3.30.1.3	calculateBackgrounds	46
3.30.1.4	calculateSource	46
3.30.1.5	calculateSources	46
3.30.1.6	forwardProjection	46
3.30.1.7	generateNormalization	47
3.30.1.8	setDimensions	47
3.30.1.9	setEdge	47
3.30.1.10	setEdge1	47
3.30.1.11	setMinCrystalInRing	47
3.30.1.12	setParams	47
3.31	less_then Struct Reference	47
3.32	ListEvent Struct Reference	48
3.33	MICBase Class Reference	48
3.34	MICChiSquare Class Reference	49
3.35	MICCollimatorPhysics Class Reference	49
3.36	MICFFT Class Reference	50
3.36.1	Member Function Documentation	50
3.36.1.1	destroyFFT	50
3.37	MICGreensFunction Class Reference	50
3.37.1	Member Function Documentation	51

3.37.1.1	greensIntegral	51
3.37.1.2	integrationGreensFunction	51
3.37.1.3	mirrorRhoField	51
3.37.1.4	multiplyCompelxFields	51
3.38	OpenCLBase Class Reference	51
3.38.1	Member Function Documentation	52
3.38.1.1	ocl_fillMemory	52
3.38.1.2	ocl_getDeviceCount	52
3.38.1.3	ocl_getDeviceName	52
3.38.1.4	ocl_getUniqueDevices	52
3.38.1.5	ocl_loadKernelFromSource	53
3.38.1.6	ocl_setDevice	53
3.39	OpenCLChiSquare Class Reference	53
3.40	OpenCLChiSquareRuntime Class Reference	53
3.40.1	Constructor & Destructor Documentation	54
3.40.1.1	OpenCLChiSquareRuntime	54
3.40.1.2	OpenCLChiSquareRuntime	54
3.40.1.3	~OpenCLChiSquareRuntime	54
3.40.2	Member Function Documentation	54
3.40.2.1	checkChiSquareKernels	54
3.40.2.2	compileProgram	54
3.40.2.3	freeChiSquare	54
3.40.2.4	initChiSquare	54
3.40.2.5	launchChiSquare	55
3.40.2.6	writeFunc	55
3.40.2.7	writeMap	55
3.40.2.8	writeParams	55
3.41	OpenCLCollimatorPhysics Class Reference	55
3.42	OpenCLFFT Class Reference	56
3.43	OpenCLGreensFunction Class Reference	56
3.43.1	Constructor & Destructor Documentation	57
3.43.1.1	OpenCLGreensFunction	57
3.43.1.2	OpenCLGreensFunction	57
3.43.1.3	~OpenCLGreensFunction	57
3.43.2	Member Function Documentation	57
3.43.2.1	buildProgram	57
3.43.2.2	greensIntegral	57
3.43.2.3	integrationGreensFunction	57
3.43.2.4	mirrorRhoField	57
3.43.2.5	multiplyCompelxFields	58

3.44 Parameter Class Reference	58
3.45 PART Struct Reference	58
3.46 Part Struct Reference	59
3.47 PART_OPENCL Struct Reference	59
3.48 PART_SMALL Struct Reference	59
3.49 RNDState Struct Reference	60
3.50 State Struct Reference	60
3.51 Vector Struct Reference	60
3.52 VoxelPosition Struct Reference	60
3.53 voxelPosition Struct Reference	61
Index	62

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

BaseFFT	5
CudaFFT	18
MICFFT	50
OpenCLFFT	56
ChiSquareRuntime	6
CudaChiSquareRuntime	14
OpenCLChiSquareRuntime	53
compare_particle	7
compare_particle_small	8
CUDA_PART2	8
CUDA_PART2_SMALL	8
CUDA_PART_SMALL	9
CudaBase	9
CudaChiSquare	14
Device	23
DKSAutoTuning	23
DKSAutoTuningTester	24
DKSBase	24
DKSFFT	35
DKSBaseMuSR	31
DKSOPAL	38
DKSImageRecon	36
DKSCollimatorPhysics	33
CudaCollimatorPhysics	16
MICCollimatorPhysics	49
OpenCLCollimatorPhysics	55
DKSConfig	34
DKSSearchStates	41
DKSStream	43
DKSTimer	43
Double3	44
GreensFunction	44
CudaGreensFunction	19
MICGreensFunction	50
OpenCLGreensFunction	56
ImageReconstruction	45

CudaImageReconstruction	20
less_then	47
ListEvent	48
MICBase	48
MICChiSquare	49
OpenCLBase	51
OpenCLChiSquare	53
Parameter	58
PART	58
Part	59
PART_OPENCL	59
PART_SMALL	59
RNDState	60
State	60
Vector	60
VoxelPosition	60
voxelPosition	61

Chapter 2

Class Index

2.1 Class List

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

BaseFFT	5
ChiSquareRuntime	6
compare_particle	7
compare_particle_small	8
CUDA_PART2	8
CUDA_PART2_SMALL	8
CUDA_PART_SMALL	9
CudaBase	9
CudaChiSquare	14
CudaChiSquareRuntime	14
CudaCollimatorPhysics	16
CudaFFT	18
CudaGreensFunction	19
CudaImageReconstruction	20
Device	23
DKSAutoTuning	23
DKSAutoTuningTester	24
DKSBase	24
DKSBaseMuSR	31
DKSCollimatorPhysics	33
DKSConfig	34
DKSFFT	35
DKSImageRecon	36
DKSOPAL	38
DKSSearchStates	41
DKSStream	43
DKSTimer	43
Double3	44
GreensFunction	44
ImageReconstruction	45
less_then	47
ListEvent	48
MICBase	48
MICChiSquare	49
MICCollimatorPhysics	49
MICFFT	50
MICGreensFunction	50
OpenCLBase	51

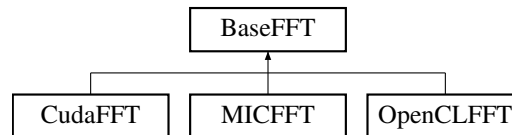
OpenCLChiSquare	53
OpenCLChiSquareRuntime	53
OpenCLCollimatorPhysics	55
OpenCLFFT	56
OpenCLGreensFunction	56
Parameter	58
PART	58
Part	59
PART_OPENCL	59
PART_SMALL	59
RNDState	60
State	60
Vector	60
VoxelPosition	60
voxelPosition	61

Chapter 3

Class Documentation

3.1 BaseFFT Class Reference

Inheritance diagram for BaseFFT:



Public Member Functions

- virtual int **setupFFT** (int ndim, int N[3])=0
- virtual int **setupFFTRC** (int ndim, int N[3], double scale=1.0)=0
- virtual int **setupFFTCCR** (int ndim, int N[3], double scale=1.0)=0
- virtual int **destroyFFT** ()=0
- virtual int **executeFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1, bool forward=true)=0
- virtual int **executeIFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1)=0
- virtual int **normalizeFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1)=0
- virtual int **executeRCFFT** (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)=0
- virtual int **executeCRFFT** (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)=0
- virtual int **normalizeCRFFT** (void *real_ptr, int ndim, int N[3], int streamId=-1)=0

Protected Member Functions

- bool **useDefaultPlan** (int ndim, int N[3])

Protected Attributes

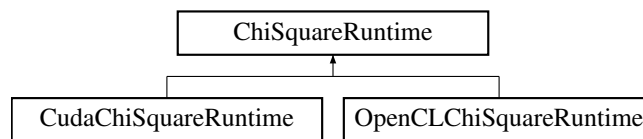
- int **defaultN** [3]
- int **defaultNdim**

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/FFT.h

3.2 ChiSquareRuntime Class Reference

Inheritance diagram for ChiSquareRuntime:



Public Member Functions

- virtual [~ChiSquareRuntime](#) ()
- virtual int **compileProgram** (std::string function, bool mlh=false)=0
- virtual int **launchChiSquare** (int fitType, void *mem_data, void *mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double &result)=0
- virtual int **writeParams** (const double *params, int numparams)=0
- virtual int **writeFunc** (const double *func, int numfunc)=0
- virtual int **writeMap** (const int *map, int nummap)=0
- virtual int **initChiSquare** (int size_data, int size_param, int size_func, int size_map)=0
- virtual int **freeChiSquare** ()=0
- virtual int **checkChiSquareKernels** (int fitType, int &threadsPerBlock)=0
- int [setConsts](#) (double N0, double tau, double bkg)
- int [setConsts](#) (double alpha, double beta)
- int [setKernelParams](#) (int numBlocks, int blockSize)
- int [getOperations](#) (int &oper)

Protected Member Functions

- void **setN0** (double value)
- void **setTau** (double value)
- void **setBKG** (double value)
- void **setAlpha** (double value)
- void **setBeta** (double value)

Protected Attributes

- double **N0_m**
- double **tau_m**
- double **bkg_m**
- double **alpha_m**
- double **beta_m**
- bool **initDone_m**
- void * **mem_chisq_m**
- void * **mem_param_m**
- void * **mem_func_m**
- void * **mem_map_m**
- int **numBlocks_m**
- int **blockSize_m**
- char * **ptx_m**

Friends

- class **DKSBaseMuSR**

3.2.1 Constructor & Destructor Documentation

3.2.1.1 virtual **ChiSquareRuntime::~ChiSquareRuntime** () [inline], [virtual]

Default constructor Default destructor

3.2.2 Member Function Documentation

3.2.2.1 int **ChiSquareRuntime::getOperations** (int & *oper*) [inline]

Get the number of operations in compiled kernel. Count the number of operation in the ptx file for the compiled program.

3.2.2.2 int **ChiSquareRuntime::setConsts** (double *N0*, double *tau*, double *bkg*) [inline]

Set N0, tau and bkg values to use for the kernel. If values changes between data sets this needs to be called before every kernel call. Returns DKS_SUCCESS.

3.2.2.3 int **ChiSquareRuntime::setConsts** (double *alpha*, double *beta*) [inline]

Set alpha and beta values to use for the kernel. If values changes between data sets this needs to be called before every kernel call. Returns DKS_SUCCESS.

3.2.2.4 int **ChiSquareRuntime::setKernelParams** (int *numBlocks*, int *blockSize*) [inline]

Set number of blocks and threads. Used to set parameters obtained from auto-tuning

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/ChiSquareRuntime.h

3.3 compare_particle Struct Reference

Public Member Functions

- void **set_threshold** (int t)
- __host__ __device__ bool **operator()** (CUDA_PART p1, CUDA_PART p2)
- __host__ __device__ bool **operator()** (CUDA_PART p1)

Public Attributes

- int **threshold**

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cu

3.4 `compare_particle_small` Struct Reference

Public Member Functions

- void **set_threshold** (int t)
- `__host__ __device__` bool **operator()** ([CUDA_PART_SMALL](#) p1, [CUDA_PART_SMALL](#) p2)
- `__host__ __device__` bool **operator()** ([CUDA_PART_SMALL](#) p1)

Public Attributes

- int **threshold**

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cu

3.5 `CUDA_PART2` Struct Reference

Public Attributes

- int * **label**
- unsigned * **localID**
- double3 * **Rincol**
- double3 * **Pincol**
- long * **IDincol**
- int * **Binincol**
- double * **DTincol**
- double * **Qincol**
- long * **LastSecincol**
- double3 * **Bfincol**
- double3 * **Efincol**

3.5.1 Detailed Description

Structure for storing particle on GPU

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cuh

3.6 `CUDA_PART2_SMALL` Struct Reference

Public Attributes

- int * **label**
- unsigned * **localID**
- double3 * **Rincol**
- double3 * **Pincol**

3.6.1 Detailed Description

Structure for storing particle on GPU

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cuh

3.7 CUDA_PART_SMALL Struct Reference

Public Attributes

- int **label**
- unsigned **localID**
- double3 **Rincol**
- double3 **Pincol**

3.7.1 Detailed Description

Structure for storing particle on GPU

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cuh

3.8 CudaBase Class Reference

Public Member Functions

- int [cuda_createCurandStates](#) (int size, int seed=-1)
- int [cuda_deleteCurandStates](#) ()
- int [cuda_createRandomNumbers](#) (void *mem_ptr, int size)
- curandState * [cuda_getCurandStates](#) ()
- int [cuda_createStream](#) (int &streamId)
- int [cuda_addStream](#) (cudaStream_t tmpStream, int &streamId)
- int [cuda_deleteStream](#) (int id)
- int [cuda_deleteStreams](#) ()
- int [cuda_setStream](#) (int id)
- int [cuda_getStreamId](#) ()
- int [cuda_defaultStream](#) ()
- int [cuda_numberOfStreams](#) ()
- cudaStream_t [cuda_getStream](#) (int id)
- cublasHandle_t [cuda_getCublas](#) ()
- int [cuda_getDevices](#) ()
- int [cuda_getDeviceCount](#) (int &ndev)
- int [cuda_getDeviceName](#) (std::string &device_name)
- int [cuda_setDevice](#) (int device)
- int [cuda_getUniqueDevices](#) (std::vector< int > &devices)
- int [cuda_setUp](#) ()
- void * [cuda_allocateMemory](#) (size_t size, int &ierr)
- template<typename T >
int [cuda_allocateHostMemory](#) (T *&ptr, size_t size)

- `template<typename T >`
`int cuda_zeroMemory (T *mem_ptr, size_t size, int offset=0)`
- `template<typename T >`
`int cuda_zeroMemoryAsync (T *mem_ptr, size_t size, int offset=0, int streamId=-1)`
- `template<typename T >`
`int cuda_writeData (T *mem_ptr, const void *in_data, size_t size, int offset=0)`
- `template<typename T >`
`int cuda_writeDataAsync (T *mem_ptr, const void *in_data, size_t size, int streamId=-1, int offset=0)`
- `template<typename T >`
`int cuda_readData (const T *mem_ptr, void *out_data, size_t size, int offset=0)`
- `template<typename T >`
`int cuda_readDataAsync (const T *mem_ptr, void *out_data, size_t size, int streamId=-1, int offset=0)`
- `int cuda_freeMemory (void *mem_ptr)`
- `int cuda_freeHostMemory (void *mem_ptr)`
- `template<typename T >`
`void * cuda_pushData (const void *in_data, size_t size, int &ierr)`
- `template<typename T >`
`int cuda_pullData (T *mem_ptr, void *out_data, size_t size, int &ierr)`
- `int cuda_executeFunction ()`
- `int cuda_cleanUp ()`
- `int cuda_syncDevice ()`
- `template<typename T >`
`int cuda_hostRegister (T *ptr, int size)`
- `template<typename T >`
`int cuda_hostUnregister (T *ptr)`
- `int cuda_memInfo ()`

Protected Attributes

- `cublasHandle_t defaultCublas`
- `curandState * defaultRndState`
- `int defaultRndSet`

3.8.1 Member Function Documentation

3.8.1.1 `int CudaBase::cuda_addStream (cudaStream_t tmpStream, int & streamId)`

add existing cuda stream to the list. Return: success or error code.

3.8.1.2 `template<typename T > int CudaBase::cuda_allocateHostMemory (T *& ptr, size_t size)` `[inline]`

Info: allocate host memory in pinned memory Return: success or error code

3.8.1.3 `void * CudaBase::cuda_allocateMemory (size_t size, int & ierr)`

Info: allocate memory on cuda device Return: pointer to memory object

3.8.1.4 `int CudaBase::cuda_cleanUp ()`

Info: clean up Return: success or error code

3.8.1.5 `int CudaBase::cuda_createCurandStates (int size, int seed = -1)`

Init cuda random number (cuRand) states. Create an array of type curandState with "size" elements on the GPU and create a curandState with different seed for each array entry. If no seed is given create a seed based on current time. Return success or error code

3.8.1.6 `int CudaBase::cuda_createRandomNumbers (void * mem_ptr, int size)`

Create 'size' random numbers on the device and save in mem_ptr array

3.8.1.7 `int CudaBase::cuda_createStream (int & streamId)`

Create a cuda stream and set streamId to index referring to this stream. Return success or error code

3.8.1.8 `int CudaBase::cuda_defaultStream ()`

Info: reset to default stream Return: success or error code

3.8.1.9 `int CudaBase::cuda_deleteCurandStates ()`

Delete curandState. Delete curandState array on the GPU and free memory. Return success or error code

3.8.1.10 `int CudaBase::cuda_deleteStream (int id)`

delete cuda stream success or error code

3.8.1.11 `int CudaBase::cuda_deleteStreams ()`

delete all streams success or error code

3.8.1.12 `int CudaBase::cuda_executeFunction ()`

Info: execute function Return: success or error code

3.8.1.13 `int CudaBase::cuda_freeHostMemory (void * mem_ptr)`

Info: free page locked memory on host Return: success or error code

3.8.1.14 `int CudaBase::cuda_freeMemory (void * mem_ptr)`

Info: free memory on device Return: success or error code

3.8.1.15 `cublasHandle_t CudaBase::cuda_getCublas ()`

Get default cublass handle

3.8.1.16 `curandState * CudaBase::cuda_getCurandStates ()`

Get a pointer to curand states

3.8.1.17 `int CudaBase::cuda_getDeviceCount (int & ndev)`

Get CUDA device count. Sets the number of devices on the platform that can use CUDA. Returns DKS_SUCCESS

3.8.1.18 `int CudaBase::cuda_getDeviceName (std::string & device_name)`

Get the name of the device. QUery the device properties of the used device and set the string *device_name*

3.8.1.19 `int CudaBase::cuda_getDevices ()`

Info: get information on cuda devices Return: success or error code

3.8.1.20 `cudaStream_t CudaBase::cuda_getStream (int id)`

Info: get stream Return: stream

3.8.1.21 `int CudaBase::cuda_getStreamId ()`

Info: get stream that is used Return: return id of curretn stream

3.8.1.22 `int CudaBase::cuda_getUniqueDevices (std::vector< int > & devices)`

Get unique devices Get array of indeces with the unique CUDA devices available on the paltform

3.8.1.23 `template<typename T> int CudaBase::cuda_hostRegister (T * ptr, int size)` `[inline]`

Page-lock host memory

3.8.1.24 `template<typename T> int CudaBase::cuda_hostUnregister (T * ptr)` `[inline]`

Release page locked memory

3.8.1.25 `int CudaBase::cuda_memInfo ()` `[inline]`

Info: print device memory info (total, used, avail) Return: success or error code

3.8.1.26 `int CudaBase::cuda_numberOfStreams ()`

Info: get number of streams Return: success or error code

3.8.1.27 `template<typename T> int CudaBase::cuda_pullData (T * mem_ptr, void * out_data, size_t size, int & ierr)`
`[inline]`

Info: read data and free memory (pull) Return: success or error code

3.8.1.28 `template<typename T> void* CudaBase::cuda_pushData (const void * in_data, size_t size, int & ierr)`
`[inline]`

Info: allcate memory and write data (push) Return: pointer to memory object

3.8.1.29 `template<typename T> int CudaBase::cuda_readData (const T * mem_ptr, void * out_data, size_t size, int offset = 0) [inline]`

Info: read data from memory Return: success or error code

3.8.1.30 `template<typename T> int CudaBase::cuda_readDataAsync (const T * mem_ptr, void * out_data, size_t size, int streamId = -1, int offset = 0) [inline]`

Info: read data async from device memory Return: success or error code

3.8.1.31 `int CudaBase::cuda_setDevice (int device)`

Set CUDA device to use. If device passed in is larger than the number of devices use the default:0 and return DKS_ERROR

3.8.1.32 `int CudaBase::cuda_setStream (int id)`

set stream to use success or error code

3.8.1.33 `int CudaBase::cuda_setUp ()`

Info: init device Return: success or error code

3.8.1.34 `int CudaBase::cuda_syncDevice () [inline]`

Info: sync cuda device Return: success or error code

3.8.1.35 `template<typename T> int CudaBase::cuda_writeData (T * mem_ptr, const void * in_data, size_t size, int offset = 0) [inline]`

Info: write data to memory Return: success or error code

3.8.1.36 `template<typename T> int CudaBase::cuda_writeDataAsync (T * mem_ptr, const void * in_data, size_t size, int streamId = -1, int offset = 0) [inline]`

Info: write data asynchronously Return: success or error code

3.8.1.37 `template<typename T> int CudaBase::cuda_zeroMemory (T * mem_ptr, size_t size, int offset = 0) [inline]`

Zero CUDA memory. Set all the elements of the array on the device to zero.

3.8.1.38 `template<typename T> int CudaBase::cuda_zeroMemoryAsync (T * mem_ptr, size_t size, int offset = 0, int streamId = -1) [inline]`

Zero CUDA memory. Set all the elements of the array on the device to zero.

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaBase.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaBase.cu

3.9 CudaChiSquare Class Reference

Public Member Functions

- [CudaChiSquare](#) ([CudaBase](#) *base)
- int **cuda_PHistoTFFcn** (void *mem_data, void *mem_par, void *mem_chisq, double fTimeResolution, double fRebin, int sensors, int length, int numpar, double &result)
- int **cuda_singleGaussTF** (void *mem_data, void *mem_t0, void *mem_par, void *mem_result, double fTimeResolution, double fRebin, double fGoodBinOffset, int sensors, int length, int numpar, double &result)
- int **cuda_doubleLorentzTF** (void *mem_data, void *mem_t0, void *mem_par, void *mem_result, double fTimeResolution, double fRebin, double fGoodBinOffset, int sensors, int length, int numpar, double &result)

3.9.1 Constructor & Destructor Documentation

3.9.1.1 CudaChiSquare::CudaChiSquare (CudaBase * base) [inline]

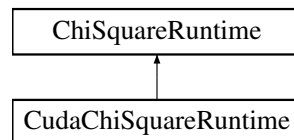
Constructor which gets [CudaBase](#) as argument

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaChiSquare.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaChiSquare.cu

3.10 CudaChiSquareRuntime Class Reference

Inheritance diagram for CudaChiSquareRuntime:



Public Member Functions

- [CudaChiSquareRuntime](#) ([CudaBase](#) *base)
- [CudaChiSquareRuntime](#) ()
- [~CudaChiSquareRuntime](#) ()
- int **compileProgram** (std::string function, bool mlh=false)
- int **launchChiSquare** (int fitType, void *mem_data, void *mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double &result)
- int **writeParams** (const double *params, int numparams)
- int **writeFunc** (const double *func, int numfunc)
- int **writeMap** (const int *map, int nummap)
- int **initChiSquare** (int size_data, int size_param, int size_func, int size_map)
- int **freeChiSquare** ()
- int **checkChiSquareKernels** (int fitType, int &threadsPerBlock)

Additional Inherited Members

3.10.1 Constructor & Destructor Documentation

3.10.1.1 CudaChiSquareRuntime::CudaChiSquareRuntime (CudaBase * *base*)

Constructor with [CudaBase](#) argument

3.10.1.2 CudaChiSquareRuntime::CudaChiSquareRuntime ()

Default constructor init cuda device

3.10.1.3 CudaChiSquareRuntime::~CudaChiSquareRuntime ()

Default destructor

3.10.2 Member Function Documentation

3.10.2.1 int CudaChiSquareRuntime::checkChiSquareKernels (int *fitType*, int & *threadsPerBlock*) [inline], [virtual]

Check if CUDA device is able to run the chi square kernel. Redundant - all new CUDA devices that support RT compilation will also support double precision, there are no other requirements to run chi square on GPU

Implements [ChiSquareRuntime](#).

3.10.2.2 int CudaChiSquareRuntime::compileProgram (std::string *function*, bool *mlh* = false) [virtual]

Compile program and save ptx. Add function string to the calcFunction kernel and compile the program Function must be valid C math expression. Parameters can be addressed in a form par[map[idx]]

Implements [ChiSquareRuntime](#).

3.10.2.3 int CudaChiSquareRuntime::freeChiSquare () [virtual]

Free temporary memory allocated for chi square. Frees the chisq temporary memory and memory for params, functions and maps

Implements [ChiSquareRuntime](#).

3.10.2.4 int CudaChiSquareRuntime::initChiSquare (int *size_data*, int *size_param*, int *size_func*, int *size_map*) [virtual]

Allocate temporary memory needed for chi square. Initializes the necessary temporary memory for the chi square calculations. Size_data needs to the maximum number of elements in any datasets that will be used for calculations. Size_param, size_func and size_map are the maximum number of parameters, functions and maps used in calculations.

Implements [ChiSquareRuntime](#).

3.10.2.5 int CudaChiSquareRuntime::launchChiSquare (int *fitType*, void * *mem_data*, void * *mem_err*, int *length*, int *numpar*, int *numfunc*, int *nummap*, double *timeStart*, double *timeStep*, double & *result*) [virtual]

Launch selected kernel Launched the selected kernel from the compiled code. Result is put in &result variable

Implements [ChiSquareRuntime](#).

3.10.2.6 `int CudaChiSquareRuntime::writeFunc (const double * func, int numfunc) [virtual]`

Write functions to device. Write function values from double array to mem_func_m memory on the device.

Implements [ChiSquareRuntime](#).

3.10.2.7 `int CudaChiSquareRuntime::writeMap (const int * map, int nummap) [virtual]`

Write maps to device. Write map values from int array to mem_map_m memory on the device.

Implements [ChiSquareRuntime](#).

3.10.2.8 `int CudaChiSquareRuntime::writeParams (const double * params, int numparams) [virtual]`

Write params to device. Write params from double array to mem_param_m memory on the device.

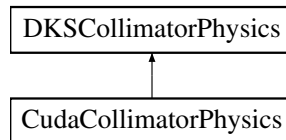
Implements [ChiSquareRuntime](#).

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaChiSquareRuntime.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaChiSquareRuntime.cu

3.11 CudaCollimatorPhysics Class Reference

Inheritance diagram for CudaCollimatorPhysics:



Public Member Functions

- [CudaCollimatorPhysics](#) ([CudaBase](#) *base)
- [CudaCollimatorPhysics](#) ()
- [~CudaCollimatorPhysics](#) ()
- [CollimatorPhysics](#) (void *mem_ptr, void *par_ptr, int numpartices, bool enableRutherfordScattering=true)
- [CollimatorPhysicsSoA](#) (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles)
- [CollimatorPhysicsSort](#) (void *mem_ptr, int numparticles, int &numaddback)
- [CollimatorPhysicsSortSoA](#) (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles, int &numaddback)
- [ParallelTrackerPush](#) (void *r_ptr, void *p_ptr, int npart, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)
- [ParallelTrackerKick](#) (void *r_ptr, void *p_ptr, void *ef_ptr, void *bf_ptr, void *dt_ptr, double charge, double mass, int npart, double c, int streamId=-1)
- [ParallelTrackerPushTransform](#) (void *x_ptr, void *p_ptr, void *lastSec_ptr, void *orient_ptr, int npart, int nsec, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)

Additional Inherited Members

3.11.1 Detailed Description

[CudaCollimatorPhysics](#) class. Contains kerenls that execute CollimatorPhysics functions form OPAL. For detailed documentation on CollimatorPhysics functions see OPAL documentation

3.11.2 Constructor & Destructor Documentation

3.11.2.1 CudaCollimatorPhysics::CudaCollimatorPhysics (CudaBase * base) [inline]

Constructor with [CudaBase](#) argument

3.11.2.2 CudaCollimatorPhysics::CudaCollimatorPhysics () [inline]

Constructor - empty.

3.11.2.3 CudaCollimatorPhysics::~~CudaCollimatorPhysics () [inline]

Destructor - empty

3.11.3 Member Function Documentation

3.11.3.1 int CudaCollimatorPhysics::CollimatorPhysics (void * mem_ptr, void * par_ptr, int numparticles, bool enableRutherfordScattering = true) [virtual]

Execute collimator physics kernel.

Implements [DKSCollimatorPhysics](#).

3.11.3.2 int CudaCollimatorPhysics::CollimatorPhysicsSort (void * mem_ptr, int numparticles, int & numadddback) [virtual]

Sort particle array on GPU. Count particles that are dead (label -1) or leaving material (label -2) and sort particle array so these particles are at the end of array

Implements [DKSCollimatorPhysics](#).

3.11.3.3 int CudaCollimatorPhysics::ParallelTTrackerPush (void * r_ptr, void * p_ptr, int npart, void * dt_ptr, double dt, double c, bool usedt = false, int streamId = -1) [virtual]

BorisPusher push function for integration from OPAL. ParallelTTracker integration from OPAL implemented in cuda. For more details see ParallelTTracker documentation in opal

Implements [DKSCollimatorPhysics](#).

3.11.3.4 int CudaCollimatorPhysics::ParallelTTrackerPushTransform (void * x_ptr, void * p_ptr, void * lastSec_ptr, void * orient_ptr, int npart, int nsec, void * dt_ptr, double dt, double c, bool usedt = false, int streamId = -1) [virtual]

BorisPusher push function with transformto function form OPAL ParallelTTracker integration from OPAL implemented in cuda. For more details see ParallelTTracker documentation in opal

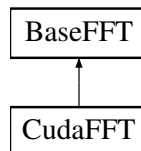
Implements [DKSCollimatorPhysics](#).

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cu

3.12 CudaFFT Class Reference

Inheritance diagram for CudaFFT:



Public Member Functions

- [CudaFFT](#) ([CudaBase](#) *base)
- [CudaFFT](#) ()
- [~CudaFFT](#) ()
- int [setupFFT](#) (int ndim, int N[3])
- int [setupFFTRC](#) (int ndim, int N[3], double scale=1.0)
- int [setupFFTCR](#) (int ndim, int N[3], double scale=1.0)
- int [destroyFFT](#) ()
- int [executeFFT](#) (void *mem_ptr, int ndim, int N[3], int streamId=-1, bool forward=true)
- int [executeIFFT](#) (void *mem_ptr, int ndim, int N[3], int streamId=-1)
- int [normalizeFFT](#) (void *mem_ptr, int ndim, int N[3], int streamId=-1)
- int [executeRCFFT](#) (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)
- int [executeCRFFT](#) (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)
- int [normalizeCRFFT](#) (void *real_ptr, int ndim, int N[3], int streamId=-1)

Additional Inherited Members

3.12.1 Constructor & Destructor Documentation

3.12.1.1 CudaFFT::CudaFFT (CudaBase * base)

Constructor with [CudaBase](#) as argument

3.12.1.2 CudaFFT::CudaFFT ()

constructor

3.12.1.3 CudaFFT::~~CudaFFT ()

destructor

3.12.2 Member Function Documentation

3.12.2.1 `int CudaFFT::destroyFFT () [virtual]`

Info: destroy default FFT plans Return: success or error code

Implements [BaseFFT](#).

3.12.2.2 `int CudaFFT::setupFFT (int ndim, int N[3]) [virtual]`

Info: init cufftPlans witch can be reused for all FFTs of the same size and type Return: success or error code

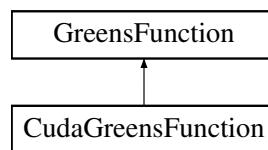
Implements [BaseFFT](#).

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaFFT.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaFFT.cu

3.13 CudaGreensFunction Class Reference

Inheritance diagram for CudaGreensFunction:



Public Member Functions

- [CudaGreensFunction](#) ([CudaBase](#) *base)
- `int greensIntegral` (void *tmpgreen, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId=-1)
- `int integrationGreensFunction` (void *rho2_m, void *tmpgreen, int I, int J, int K, int streamId=-1)
- `int mirrorRhoField` (void *rho2_m, int I, int J, int K, int streamId=-1)
- `int multiplyCompelxFields` (void *ptr1, void *ptr2, int size, int streamId=-1)

3.13.1 Constructor & Destructor Documentation

3.13.1.1 `CudaGreensFunction::CudaGreensFunction (CudaBase * base)`

Constructor with [CudaBase](#) argument

3.13.2 Member Function Documentation

3.13.2.1 `int CudaGreensFunction::greensIntegral (void * tmpgreen, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId = -1) [virtual]`

Info: calc itegral on device memory (taken from OPAL src code) Return: success or error code

Implements [GreensFunction](#).

3.13.2.2 `int CudaGreensFunction::integrationGreensFunction (void * rho2_m, void * tmpgreen, int I, int J, int K, int streamId = -1) [virtual]`

Info: integration of rho2_m field (taken from OPAL src code) Return: success or error code

Implements [GreensFunction](#).

3.13.2.3 `int CudaGreensFunction::mirrorRhoField (void * rho2_m, int I, int J, int K, int streamId = -1) [virtual]`

Info: mirror rho field (taken from OPAL src code) Return: succes or error code

Implements [GreensFunction](#).

3.13.2.4 `int CudaGreensFunction::multiplyCompelxFields (void * ptr1, void * ptr2, int size, int streamId = -1) [virtual]`

Info: multiply complex fields already on the GPU memory, result will be put in ptr1 Return: success or error code

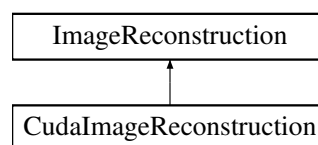
Implements [GreensFunction](#).

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaGreensFunction.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaGreensFunction.cu

3.14 CudalImageReconstruction Class Reference

Inheritance diagram for CudalImageReconstruction:



Public Member Functions

- [CudalImageReconstruction](#) ()
- [CudalImageReconstruction](#) ([CudaBase](#) *base)
- [~CudalImageReconstruction](#) ()
- [calculateSource](#) (void *image_space, void *image_position, void *source_position, void *avg, void *std, float diameter, int total_voxels, int total_sources, int start=0)
- [calculateBackground](#) (void *image_space, void *image_position, void *source_position, void *avg, void *std, float diameter, int total_voxels, int total_sources, int start=0)
- [calculateSources](#) (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)
- [calculateBackgrounds](#) (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)
- [generateNormalization](#) (void *recon, void *image_position, void *det_position, int total_det)
- [forwardProjection](#) (void *correction, void *recon, void *list_data, void *det_position, void *image_position, int num_events)
- [backwardProjection](#) (void *correction, void *recon_corrector, void *list_data, void *det_position, void *image_position, int num_events, int num_voxels)
- [setDimensions](#) (int voxel_x, int voxel_y, int voxel_z, float voxel_size)

- int [setEdge](#) (float x_edge, float y_edge, float z_edge)
- int [setEdge1](#) (float x_edge1, float y_edge1, float z_edge1, float z_edge2)
- int [setMinCrystalInRing](#) (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1)
- int [setParams](#) (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter)

Additional Inherited Members

3.14.1 Constructor & Destructor Documentation

3.14.1.1 CudalImageReconstruction::CudalImageReconstruction () [inline]

Constructor

3.14.1.2 CudalImageReconstruction::CudalImageReconstruction (CudaBase * base) [inline]

Constructor with base

3.14.1.3 CudalImageReconstruction::~~CudalImageReconstruction () [inline]

Destructor

3.14.2 Member Function Documentation

3.14.2.1 int CudalImageReconstruction::backwardProjection (void * correction, void * recon_corrector, void * list_data, void * det_position, void * image_position, int num_events, int num_voxels) [virtual]

Calculate backward projection. For image reconstruction calculates backward projections. see recon.cpp for details
Implements [ImageReconstruction](#).

3.14.2.2 int CudalImageReconstruction::calculateBackground (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0) [virtual]

Cuda implementation of calculate background

Implements [ImageReconstruction](#).

3.14.2.3 int CudalImageReconstruction::calculateBackgrounds (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0) [virtual]

Calculate background for different sources

Implements [ImageReconstruction](#).

3.14.2.4 int CudalImageReconstruction::calculateSource (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0) [virtual]

CUDA implementation of calculate source

Implements [ImageReconstruction](#).

3.14.2.5 `int CudalImageReconstruction::calculateSources (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0)` [virtual]

Caluculate source for differente sources

Implements [ImageReconstruction](#).

3.14.2.6 `int CudalImageReconstruction::forwardProjection (void * correction, void * recon, void * list_data, void * det_position, void * image_position, int num_events)` [virtual]

Calculate forward projection. For image reconstruction calculates forward projections. see recon.cpp for details

Implements [ImageReconstruction](#).

3.14.2.7 `int CudalImageReconstruction::generateNormalization (void * recon, void * image_position, void * det_position, int total_det)` [virtual]

Generate normalization. Goes trough detectors pairs and if detector pair crosses image launches separte kernel that updates voxel values in the image on the slope between these two detectors.

Implements [ImageReconstruction](#).

3.14.2.8 `int CudalImageReconstruction::setDimensions (int voxel_x, int voxel_y, int voxel_z, float voxel_size)` [virtual]

Set the voxel dimensins on device.

Implements [ImageReconstruction](#).

3.14.2.9 `int CudalImageReconstruction::setEdge (float x_edge, float y_edge, float z_edge)` [virtual]

Set the image edge.

Implements [ImageReconstruction](#).

3.14.2.10 `int CudalImageReconstruction::setEdge1 (float x_edge1, float y_edge1, float z_edge1, float z_edge2)` [virtual]

Set the image edge1.

Implements [ImageReconstruction](#).

3.14.2.11 `int CudalImageReconstruction::setMinCrystallnRing (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1)` [virtual]

Set the minimum cristan in one ring values.

Implements [ImageReconstruction](#).

3.14.2.12 `int CudalImageReconstruction::setParams (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter)` [virtual]

Set all other required parameters for reconstruction.

Implements [ImageReconstruction](#).

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudalImageReconstruction.cuh
- /home/l_locans/gitwork/DKS-src/src/CUDA/CudalImageReconstruction.cu

3.15 Device Class Reference

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

- /home/l_locans/gitwork/DKS-src/src/DKSDevice.h

3.16 DKSAutoTuning Class Reference

Public Member Functions

- [DKSAutoTuning](#) ([DKSBase](#) *base, std::string api, std::string device, int loops=100)
- [~DKSAutoTuning](#) ()
- void [setFunction](#) (std::function< int()> f, std::string name, bool evaluate_time=true)
- void [setFunction](#) (std::function< double()> f, std::string name, bool evaluate_time=false)
- template<typename T1 >
void [addParameter](#) (T1 *value, T1 min, T1 max, T1 step, std::string name)
- void [clearParameters](#) ()
- void [exhaustiveSearch](#) ()
- void [lineSearch](#) ()
- void [hillClimbing](#) (int restart_loops=1)
- void [simulatedAnnealing](#) (double Tstart, double Tstep)

3.16.1 Constructor & Destructor Documentation

3.16.1.1 [DKSAutoTuning::DKSAutoTuning](#) ([DKSBase](#) * *base*, std::string *api*, std::string *device*, int *loops* = 100)

Constructor

3.16.1.2 [DKSAutoTuning::~~DKSAutoTuning](#) ()

Destructor

3.16.2 Member Function Documentation

3.16.2.1 template<typename T1 > void [DKSAutoTuning::addParameter](#) (T1 * *value*, T1 *min*, T1 *max*, T1 *step*, std::string *name*) [inline]

Set parameter for auto tuning. Provide a pointer to a parameter that will be changed during auto-tuning and a min-max value for this element

3.16.2.2 void [DKSAutoTuning::clearParameters](#) ()

Delete all added parameters

3.16.2.3 void [DKSAutoTuning::exhaustiveSearch](#) ()

Perform exhaustive search evaluating all the parameter configurations

3.16.2.4 void DKSAutoTuning::hillClimbing (int *restart_loops* = 1)

Perform hill climbing

3.16.2.5 void DKSAutoTuning::lineSearch ()

Perform auto-tuning. Perform line-search auto-tuning by varying parameters one at a time and keeping other parameters constant.

3.16.2.6 void DKSAutoTuning::setFunction (std::function< int()> *f*, std::string *name*, bool *evaluate_time* = true) [inline]

Set function to auto tune. Caller of setFunction is responsible to bind the correct parameters to the function with std::bind.

3.16.2.7 void DKSAutoTuning::simulatedAnnealing (double *Tstart*, double *Tstep*)

Perfor simulated annealing to find the parameters

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

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSAutoTuning.h
- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSAutoTuning.cpp

3.17 DKSAutoTuningTester Class Reference

Public Member Functions

- double **peaksZ** ()

Friends

- class **DKSBaseMuSR**

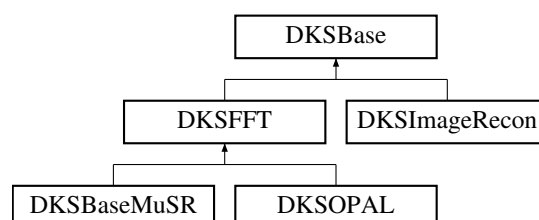
The documentation for this class was generated from the following file:

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSAutoTuningTester.h

3.18 DKSBase Class Reference

```
#include <DKSBase.h>
```

Inheritance diagram for DKSBase:



Public Member Functions

- [DKSSBase](#) ()
- [DKSSBase](#) (const char *api_name, const char *device_name)
- [~DKSSBase](#) ()
- int [setUpDevice](#) ()
- void [setAutoTuningOn](#) ()
- void [setAutoTuningOff](#) ()
- bool [isAutoTuningOn](#) ()
- void [setUseConfigOn](#) ()
- void [setUseConfigOff](#) ()
- bool [isUseConfigOn](#) ()
- int [setDevice](#) (const char *device_name, int length=-1)
- int [setAPI](#) (const char *api_name, int length=-1)
- int [getDevices](#) ()
- int [getDeviceCount](#) (int &ndev)
- int [getDeviceName](#) (std::string &device_name)
- int [setDefaultDevice](#) (int device)
- int [getDeviceList](#) (std::vector< int > &devices)
- int [initDevice](#) ()
- int [createStream](#) (int &streamId)
- int [closeHandle](#) (void *mem_ptr)
- int [syncDevice](#) ()
- template<typename T >
void * [pushData](#) (const void *data_in, int elements, int &ierr)
- template<typename T >
int [pullData](#) (void *mem_ptr, void *data_out, int elements)
- template<typename T >
void * [allocateMemory](#) (int elements, int &ierr)
- template<typename T >
int [allocateHostMemory](#) (T *&ptr, int size)
- template<typename T >
int [freeHostMemory](#) (T *&ptr, int size)
- template<typename T >
int [registerHostMemory](#) (T *ptr, int size)
- template<typename T >
int [unregisterHostMemory](#) (T *ptr)
- template<typename T >
int [writeData](#) (void *mem_ptr, const void *data, int elements, int offset=0)
- template<typename T >
int [writeDataAsync](#) (void *mem_ptr, const void *data, int elements, int streamId=-1, int offset=0)
- template<typename T >
int [readData](#) (const void *mem_ptr, void *out_data, int elements, int offset=0)
- template<typename T >
int [readDataAsync](#) (const void *mem_ptr, void *out_data, int elements, int streamId=-1, int offset=0)
- template<typename T >
int [freeMemory](#) (void *mem_ptr, int elements)
- int [callCreateRandomNumbers](#) (void *mem_ptr, int size)
- int [callInitRandoms](#) (int size, int seed=-1)
- int [callMemInfo](#) ()
- void [oclEventInfo](#) ()
- void [oclClearEvents](#) ()

Protected Member Functions

- bool [apiOpenCL](#) ()
- bool [apiCuda](#) ()
- bool [apiOpenMP](#) ()
- bool [deviceGPU](#) ()
- bool [deviceCPU](#) ()
- bool [deviceMIC](#) ()
- int [loadOpenCLKernel](#) (const char *kernel_name)
- std::string [getAPI](#) ()
- std::string [getDevice](#) ()

Protected Attributes

- [DKSConfig](#) [dksconfig](#)

3.18.1 Detailed Description

[DKSBase](#) class. [DKSBase.h](#) Author: Uldis Locans Date: 15.09.2014 Base class of Dynamic Kernel Scheduler that handles the function calls from host application to DKSDKSBase class for handling function calls to DKS library

3.18.2 Constructor & Destructor Documentation

3.18.2.1 DKSBase::DKSBase ()

Default constructor.

3.18.2.2 DKSBase::DKSBase (const char * *api_name*, const char * *device_name*)

Constructor that sets api and devcie to use with DKS.

3.18.2.3 DKSBase::~~DKSBase ()

Destructor. Free DKS resources.

3.18.3 Member Function Documentation

3.18.3.1 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. Returns succes or error code. TODO: opencl and mic implementations needed.

3.18.3.2 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.

3.18.3.3 bool DKSBase::apiCuda () [protected]

Check if current API is set to CUDA. Return true/false wether curretn api is cuda

3.18.3.4 `bool DKSBBase::apiOpenCL () [protected]`

Check if current API is set to OpenCL. Return true/false whether current api is opencl

3.18.3.5 `bool DKSBBase::apiOpenMP () [protected]`

Check if current API is set to OpenMP. Return true/false whether current api is OpenMP

3.18.3.6 `int DKSBBase::callCreateRandomNumbers (void * mem_ptr, int size)`

Create random numbers on the device and fill mem_data array

3.18.3.7 `int DKSBBase::callInitRandoms (int size, int seed = -1)`

Init random number states and save for reuse on device. If seed is -1, a random seed based on current time is taken. TODO: opencl and mic implementations.

3.18.3.8 `int DKSBBase::callMemInfo () [inline]`

Print memory information on device (total, used, available) TODO: opencl and mic implementation

3.18.3.9 `int DKSBBase::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. Together 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.

3.18.3.10 `int DKSBBase::createStream (int & streamId)`

Create stream for async execution. Function to create different streams with device to allow async 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.

3.18.3.11 `bool DKSBBase::deviceCPU () [protected]`

Check if device is CPU

3.18.3.12 `bool DKSBBase::deviceGPU () [protected]`

Check if device is GPU

3.18.3.13 `bool DKSBBase::deviceMIC () [protected]`

Check if device is MIC

3.18.3.14 `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 allocate-HostMemory. *ptr* is the host pointer where page-locked memory was allocated, *size* - number of elements held by the memroy.

3.18.3.15 `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.

3.18.3.16 `int DKSBase::getDeviceCount (int & ndev)`

Returns device count. Saves the number of the devices available on the platform to *ndev*.

3.18.3.17 `int DKSBase::getDeviceList (std::vector< int > & devices)`

Get unique devices. Get a list of all the unique devices available on the platform. When API and device type for DKS is set, *getDeviceList* can get all the unique devices available for this API and device type. Used for autotuning if multiple different GPUs are installed on the system.

3.18.3.18 `int DKSBase::getDeviceName (std::string & device_name)`

Get the name of the device in use. Query the device that is used and get the naem of the device. The name is saved in the *device_name* string. Returns DKS_SUCCESS

3.18.3.19 `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. Length specifies the number of characters in *api_name* array Returns success or error code

3.18.3.20 `int DKSBase::initDevice ()`

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

3.18.3.21 `bool DKSBase::isAutoTuningOn () [inline]`

Get status of auto tuning

3.18.3.22 `bool DKSBase::isUseConfigOn () [inline]`

Check if using config file

3.18.3.23 `int DKSBase::loadOpenCLKernel (const char * kernel_name) [protected]`

Get cbase pointerCall OpenCL base to load specified kenrel file.

3.18.3.24 void DKSBBase::oclClearEvents () [inline]

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

3.18.3.25 void DKSBBase::oclEventInfo () [inline]

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

3.18.3.26 template<typename T> int DKSBBase::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.

3.18.3.27 template<typename T> void* DKSBBase::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 transferred 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.

3.18.3.28 template<typename T> int DKSBBase::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, NI - 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, NI - 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,idx,idxz - 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.

3.18.3.29 template<typename T> int DKSBBase::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).

3.18.3.30 `template<typename T> int DKSBASE::registerHostMemory (T * ptr, int size) [inline]`

Page lock allocated host memory. Page locked memory improves data transfer between host and device (true for cuda and opencl, maybe also mic). ptr - pointer to memory that needs to be page locked, size - number of elements in array. TODO: mic and opencl implementations needed

3.18.3.31 `int DKSBASE::setAPI (const char * api_name, int length = -1)`

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. Length specifies the number of characters in api_name array (length - deprecated).

3.18.3.32 `void DKSBASE::setAutoTuningOff () [inline]`

Turn of auto tuning

3.18.3.33 `void DKSBASE::setAutoTuningOn () [inline]`

Turn on auto tuning

3.18.3.34 `int DKSBASE::setDefaultDevice (int device)`

Set the device to use. Pass the index of the device to use by dks.

3.18.3.35 `int DKSBASE::setDevice (const char * device_name, int length = -1)`

Set device to use with DKS. Sets specific device to use with DKS. Supported devices are -gpu and -mic. Length specifies the number of characters in device_name array (length - deprecated). Return success or error code.

3.18.3.36 `int DKSBASE::setupDevice ()`

Function to initialize objects based on the device used.

3.18.3.37 `void DKSBASE::setUseConfigOff () [inline]`

Turn off use of config file

3.18.3.38 `void DKSBASE::setUseConfigOn () [inline]`

Turn on use of config file

3.18.3.39 `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 which 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

3.18.3.40 `template<typename T> int DKSBaMuSR::unregisterHostMemory (T * ptr) [inline]`

Unregister page locked memory. TODO: opencl and mic implementations needed.

3.18.3.41 `template<typename T> int DKSBaMuSR::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 specify 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.

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

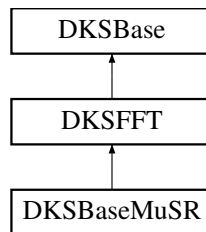
Write data to device using async write. Queue a async data write and return control to host immediately. mem_ptr - device memory pointer, data - host memory pointer, elements - number of data elements to write streamId - stream id to use, offset - offset on device from first element For trully async execution on cuda stream other than default needs to be created and device memory must be page-locked. Otherwise functions just asynchronously 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/gitwork/DKS-src/src/DKSBaMuSR.h
- /home/l_locans/gitwork/DKS-src/src/DKSBaMuSR.cpp

3.19 DKSBaMuSR Class Reference

Inheritance diagram for DKSBaMuSR:



Public Member Functions

- `int callCompileProgram` (std::string function, bool mlh=false)
- `int callLaunchChiSquare` (int fitType, void *mem_data, void *mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double &result)
- `int callAutoTuningChiSquare` (int fitType, void *mem_data, void *mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double &result, std::vector< int > &config)
- `int callSetConsts` (double N0, double tau, double bkg)
- `int callSetConsts` (double alpha, double beta)
- `int initChiSquare` (int size_data, int size_param, int size_func, int size_map)
- `int freeChiSquare` ()
- `int writeParams` (const double *params, int numparams)
- `int writeFunctions` (const double *func, int numfunc)
- `int writeMaps` (const int *map, int numfunc)
- `int checkMuSRKernels` (int fitType)
- `int checkMuSRKernels` (int fitType, int &threadsPerBlock)

- int [testAutoTuning](#) ()
- int [getOperations](#) (int &oper)

Additional Inherited Members

3.19.1 Member Function Documentation

3.19.1.1 int `DKSBaseMuSR::callAutoTuningChiSquare (int fitType, void * mem_data, void * mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double & result, std::vector< int > & config)`

Launch auto-tuning of chisquare function for the selected device. Creates a function pointer to callLaunchChiSquare with necessary arguments bind to function call. CUDA and OpenCL version - gives AutoTuning class access to numThreads parameter which is varied to find the optimal value by AutoTuning class. Uses brute force method to test all the values.

3.19.1.2 int `DKSBaseMuSR::callCompileProgram (std::string function, bool mlh = false)`

Compile the program with kernels to be run. String function contains the string that will be added to the code to compile in the function: **device** double fTheory(double t, double *p, double *f, int *m); Function string must be a valid C math expression. It can contain operators, math functions and predefined functions listed in: http://lmu.web.psi.ch/musrfit/user/MUSR/MusrFit.html#A_4.3_The_THEORY-_Block Predefined functions can be accessed by the abbreviation given in the table Parameters can be accessed in form p[idx] or p[m[idx]] - where p represents parameter array m represents map array and idx is the index to use from the maps. Precalculated function values can be accessed the same way - f[idx] or f[m[idx]]. Returns DKS_SUCCESS if everything runs successfully, otherwise returns DKS_ERROR. If DKS is compiled with debug flag enabled prints DKS error message in case something fails

3.19.1.3 int `DKSBaseMuSR::callLaunchChiSquare (int fitType, void * mem_data, void * mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double & result)`

Launch chi square calculation on data set written in mem_data memory on device. mem_par, mem_map and mem_func hold pointers to parameter, function and map values for this data set (parameter array is one for all the data sets, maps and functions change between data sets). Resulting chi square value for this dataset will be put in result variable. Returns DKS_SUCCESS if everything runs successfully, otherwise returns DKS_ERROR. If DKS is compiled with debug flag enabled prints DKS error message in case something fails

3.19.1.4 int `DKSBaseMuSR::callSetConsts (double N0, double tau, double bkg)`

Set N0, tau and BKG values for the run. Needs to be called before kernel launch if these values are changing

3.19.1.5 int `DKSBaseMuSR::callSetConsts (double alpha, double beta)`

Set alpha and beta values for the run. Needs to be called before kernel launch if these values are changing

3.19.1.6 int `DKSBaseMuSR::checkMuSRKernels (int fitType)`

Check if device can run necessary kernels. Check selected device properties to see if device supports double precision and if device can run the necessary number of work_items / work_groups to successfully execute CUDA/OpenCL kernels.

3.19.1.7 `int DKSBaSeMuSR::checkMuSRKernels (int fitType, int & threadsPerBlock)`

Perform the same check as `checkMuSRKernels(int fitType)` and return max threads per block. Used for autotuning to check what is the device limit for threads per block to correctly set the upper bound when searching the parameter space.

3.19.1.8 `int DKSBaSeMuSR::freeChiSquare ()`

Free temporary device storage allocated for χ^2 kernel. Return error code if freeing the device fails.

3.19.1.9 `int DKSBaSeMuSR::getOperations (int & oper)`

Get the number of operations in compiled kernel.

3.19.1.10 `int DKSBaSeMuSR::initChiSquare (int size_data, int size_param, int size_func, int size_map)`

Init chisquare calculations. Size is the maximum number of elements in any of the data sets used.

3.19.1.11 `int DKSBaSeMuSR::testAutoTuning ()`

Debug function to test auto-tuning search functions

3.19.1.12 `int DKSBaSeMuSR::writeFunctions (const double * func, int numfunc)`

Write function values to device. Write precalculated function values to device, memory for functions on device is handled by DKS.

3.19.1.13 `int DKSBaSeMuSR::writeMaps (const int * map, int numfunc)`

Write map indexes to device. Write map indexes to use in defined theory function to device. Memory for map indexes is handled by DKS.

3.19.1.14 `int DKSBaSeMuSR::writeParams (const double * params, int numparams)`

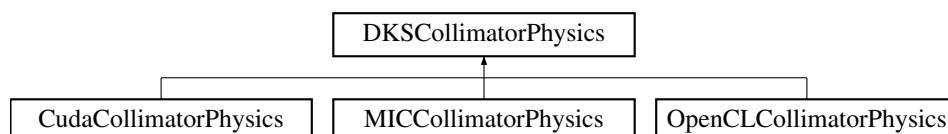
Write params to device. Write params from double array to device, params device memory is managed by DKS.

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

- `/home/l_locans/gitwork/DKS-src/src/DKSBaSeMuSR.h`
- `/home/l_locans/gitwork/DKS-src/src/DKSBaSeMuSR.cpp`

3.20 DKSCollimatorPhysics Class Reference

Inheritance diagram for DKSCollimatorPhysics:



Public Member Functions

- virtual int **CollimatorPhysics** (void *mem_ptr, void *par_ptr, int numpartices, bool enableRutherfordScattering=true)=0
- virtual int **CollimatorPhysicsSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles)=0
- virtual int **CollimatorPhysicsSort** (void *mem_ptr, int numparticles, int &numaddback)=0
- virtual int **CollimatorPhysicsSortSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles, int &numaddback)=0
- virtual int **ParallelTTrackerPush** (void *r_ptr, void *p_ptr, int npart, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)=0
- virtual int **ParallelTTrackerKick** (void *r_ptr, void *p_ptr, void *ef_ptr, void *bf_ptr, void *dt_ptr, double charge, double mass, int npart, double c, int streamId=-1)=0
- virtual int **ParallelTTrackerPushTransform** (void *x_ptr, void *p_ptr, void *lastSec_ptr, void *orient_ptr, int npart, int nsec, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)=0

Protected Attributes

- int **numBlocks_m**
- int **blockSize_m**

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/CollimatorPhysics.h

3.21 DKSConfig Class Reference

Public Member Functions

- [DKSConfig](#) ()
- int [loadConfigFile](#) ()
- int [saveConfigFile](#) ()
- int [addConfigParameter](#) (const std::string api, const std::string device, const std::string name, const std::string func, int size, std::string param, int value)
- int [getConfigParameter](#) (const std::string api, const std::string device, const std::string name, const std::string func, int size, std::string param, int &value)

3.21.1 Constructor & Destructor Documentation

3.21.1.1 DKSConfig::DKSConfig ()

Constructor, set home variable. If home directory is not set config file can not be read or saved

3.21.2 Member Function Documentation

3.21.2.1 int DKSConfig::addConfigParameter (const std::string *api*, const std::string *device*, const std::string *name*, const std::string *func*, int *size*, std::string *param*, int *value*)

Add config parameter to tree

3.21.2.2 `int DKConfig::getConfigParameter (const std::string api, const std::string device, const std::string name, const std::string func, int size, std::string param, int & value)`

Get config parameter from the tree

3.21.2.3 `int DKConfig::loadConfigFile ()`

Load autotuning.xml into tree variable if this file exists

3.21.2.4 `int DKConfig::saveConfigFile ()`

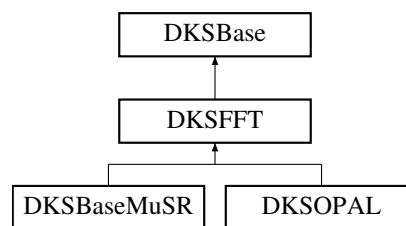
Save autotuning.xml file

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

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKConfig.h
- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKConfig.cpp

3.22 DKSFFT Class Reference

Inheritance diagram for DKSFFT:



Public Member Functions

- `int setupFFT (int ndim, int N[3])`
- `int setupFFTRC (int ndim, int N[3], double scale=1.0)`
- `int setupFFTCR (int ndim, int N[3], double scale=1.0)`
- `int callFFT (void *data_ptr, int ndim, int dimsize[3], int streamId=-1)`
- `int callIFFT (void *data_ptr, int ndim, int dimsize[3], int streamId=-1)`
- `int callNormalizeFFT (void *data_ptr, int ndim, int dimsize[3], int streamId=-1)`
- `int callR2CFFT (void *real_ptr, void *comp_ptr, int ndim, int dimsize[3], int streamId=-1)`
- `int callC2RFFT (void *real_ptr, void *comp_ptr, int ndim, int dimsize[3], int streamId=-1)`
- `int callNormalizeC2RFFT (void *real_ptr, int ndim, int dimsize[3], int streamId=-1)`

Additional Inherited Members

3.22.1 Member Function Documentation

3.22.1.1 `int DKSFFT::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_ptr* - points to complex data, *ndim* - dimension of data, *dimsize* size of each dimension. *real_ptr* size should be *dimsize*[0]**dimsize*[1]**dimsize*[2], *comp_ptr* size should be atleast (*dimsize*[0]/2+1)**dimsize*[1]**dimsize*[2] TODO: opencl and mic implementations.

3.22.1.2 `int DKSFFT::callFFT (void * data_ptr, int ndim, int dimsize[3], int streamId = -1)`

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

3.22.1.3 `int DKSFFT::callIFFT (void * data_ptr, int ndim, int dimsize[3], int streamId = -1)`

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

3.22.1.4 `int DKSFFT::callNormalizeC2RFFT (void * real_ptr, int ndim, int dimsize[3], int streamId = -1)`

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

3.22.1.5 `int DKSFFT::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.

3.22.1.6 `int DKSFFT::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_ptr* points to complex data, *ndim* - dimension of data, *dimsize* size of each dimension. *real_ptr* size should be *dimsize*[0]**dimsize*[1]**dimsize*[2], *comp_ptr* size should be at least (*dimsize*[0]/2+1)**dimsize*[1]**dimsize*[2] TODO: opengl and mic implementations

3.22.1.7 `int DKSFFT::setupFFT (int ndim, int N[3])`

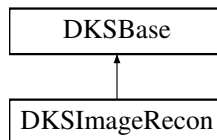
Setup FFT function. Initializes parameters for fft execution. If *ndim* > 0 initializes handles for fft calls. If ffts of various sizes are needed *setupFFT* should be called with *ndim* 0, in this case each fft will do its own setup according to fft size and dimensions. TODO: opengl and mic implementations

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

- /home/l_locans/gitwork/DKS-src/src/DKSFFT.h
- /home/l_locans/gitwork/DKS-src/src/DKSFFT.cpp

3.23 DKSIImageRecon Class Reference

Inheritance diagram for DKSIImageRecon:



Public Member Functions

- int [callCalculateSource](#) (void **image_space*, void **image_position*, void **source_position*, void **avg*, void **std*, float *diameter*, int *total_voxels*, int *total_sources*, int *start*=0)

- `int callCalculateBackground` (void *image_space, void *image_position, void *source_position, void *avg, void *std, float diameter, int total_voxels, int total_sources, int start=0)
- `int callCalculateSources` (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)
- `int callCalculateBackgrounds` (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)
- `int callGenerateNormalization` (void *recon, void *image_position, void *det_position, int total_det)
- `int callForwardProjection` (void *correction, void *recon, void *list_data, void *det_position, void *image_position, int num_events)
- `int callBackwardProjection` (void *correction, void *recon_corrector, void *list_data, void *det_position, void *image_position, int num_events, int num_voxels)
- `int setDimensions` (int voxel_x, int voxel_y, int voxel_z, float voxel_size)
- `int setEdge` (float x_edge, float y_edge, float z_edge)
- `int setEdge1` (float x_edge1, float y_edge1, float z_edge1, float z_edge2)
- `int setMinCrystalInRing` (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1)
- `int setParams` (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter)

Additional Inherited Members

3.23.1 Member Function Documentation

3.23.1.1 `int DKSIImageRecon::callBackwardProjection (void * correction, void * recon_corrector, void * list_data, void * det_position, void * image_position, int num_events, int num_voxels)`

Image reconstruction - backward projection.

3.23.1.2 `int DKSIImageRecon::callCalculateBackground (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0)`

Image reconstruction analysis calculate source.

3.23.1.3 `int DKSIImageRecon::callCalculateBackgrounds (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0)`

Image reconstruction analysis calculate source.

3.23.1.4 `int DKSIImageRecon::callCalculateSource (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0)`

Image reconstruction analysis calculate source.

3.23.1.5 `int DKSIImageRecon::callCalculateSources (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0)`

Image reconstruction analysis calculate source.

3.23.1.6 `int DKSIImageRecon::callForwardProjection (void * correction, void * recon, void * list_data, void * det_position, void * image_position, int num_events)`

Image reconstruction - forward correction.

3.23.1.7 `int DKSIImageRecon::callGenerateNormalization (void * recon, void * image_position, void * det_position, int total_det)`

Image reconstruction - generate normalization.

3.23.1.8 `int DKSIImageRecon::setDimensions (int voxel_x, int voxel_y, int voxel_z, float voxel_size)`

Set the voxel dimensins on device. Values are stored in GPU memory and used in forward and backward projection calculations. Call set function once to transfer the values from host side to GPU. If value changes on the host side set functions needs to be called again to update GPU values.

3.23.1.9 `int DKSIImageRecon::setEdge (float x_edge, float y_edge, float z_edge)`

Set the image edge. Values are stored in GPU memory and used in forward and backward projection calculations. Call set function once to transfer the values from host side to GPU. If value changes on the host side set functions needs to be called again to update GPU values.

3.23.1.10 `int DKSIImageRecon::setEdge1 (float x_edge1, float y_edge1, float z_edge1, float z_edge2)`

Set the image edge1. Values are stored in GPU memory and used in forward and backward projection calculations. Call set function once to transfer the values from host side to GPU. If value changes on the host side set functions needs to be called again to update GPU values.

3.23.1.11 `int DKSIImageRecon::setMinCrystallnRing (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1)`

Set the minimum cristan in one ring values. Values are stored in GPU memory and used in forward and backward projection calculations. Call set function once to transfer the values from host side to GPU. If value changes on the host side set functions needs to be called again to update GPU values.

3.23.1.12 `int DKSIImageRecon::setParams (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter)`

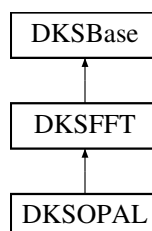
Set all other required parameters for reconstruction. Values are stored in GPU memory and used in forward and backward projection calculations. Call set function once to transfer the values from host side to GPU. If value changes on the host side set functions needs to be called again to update GPU values.

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

- /home/l_locans/gitwork/DKS-src/src/DKSIImageReconstruction.h
- /home/l_locans/gitwork/DKS-src/src/DKSIImageReconstruction.cpp

3.24 DKSOPAL Class Reference

Inheritance diagram for DKSOPAL:



Public Member Functions

- **DKSOPAL** (const char *api_name, const char *device_name)
- int **initDevice** ()
- 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)
- int **callGreensIntegration** (void *mem_ptr, void *tmp_ptr, int I, int J, int K, int streamId=-1)
- int **callMirrorRhoField** (void *mem_ptr, int I, int J, int K, int streamId=-1)
- int **callMultiplyComplexFields** (void *mem_ptr1, void *mem_ptr2, int size, int streamId=-1)
- int **callCollimatorPhysics** (void *mem_ptr, void *par_ptr, int numparticles, int numparams, int &numaddback, int &numdead, bool enableRutherfordScattering=true)
- int **callCollimatorPhysics2** (void *mem_ptr, void *par_ptr, int numparticles, bool enableRutherfordScattering=true)
- int **callCollimatorPhysicsSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles)
- int **callCollimatorPhysicsSort** (void *mem_ptr, int numparticles, int &numaddback)
- int **callCollimatorPhysicsSortSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles, int &numaddback)
- int **callParallelTTrackerPush** (void *r_ptr, void *p_ptr, int npart, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)
- int **callParallelTTrackerPushTransform** (void *x_ptr, void *p_ptr, void *lastSec_ptr, void *orient_ptr, int npart, int nsec, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)
- int **callParallelTTrackerPush** (void *r_ptr, void *p_ptr, void *dt_ptr, int npart, double c, int streamId=-1)
- int **callParallelTTrackerKick** (void *r_ptr, void *p_ptr, void *ef_ptr, void *bf_ptr, void *dt_ptr, double charge, double mass, int npart, double c, int streamId=-1)

Additional Inherited Members

3.24.1 Member Function Documentation

3.24.1.1 int DKSOPAL::callCollimatorPhysics (void * mem_ptr, void * par_ptr, int numparticles, int numparams, int & numaddback, int & numdead, bool enableRutherfordScattering = true)

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class documentation. TODO: opencl and mic implementations.

3.24.1.2 int DKSOPAL::callCollimatorPhysics2 (void * mem_ptr, void * par_ptr, int numparticles, bool enableRutherfordScattering = true)

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class documentation. TODO: opencl and mic implementations.

3.24.1.3 int DKSOPAL::callCollimatorPhysicsSoA (void * label_ptr, void * localID_ptr, void * rx_ptr, void * ry_ptr, void * rz_ptr, void * px_ptr, void * py_ptr, void * pz_ptr, void * par_ptr, int numparticles)

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class documentation. Test function for the MIC to test SoA layout vs AoS layout used in previous versions

3.24.1.4 int DKSOPAL::callCollimatorPhysicsSort (void * mem_ptr, int numparticles, int & numaddback)

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class documentation. TODO: opencl and mic implementations.

3.24.1.5 `int DKSOPAL::callCollimatorPhysicsSortSoA (void * label_ptr, void * localID_ptr, void * rx_ptr, void * ry_ptr, void * rz_ptr, void * px_ptr, void * py_ptr, void * pz_ptr, void * par_ptr, int numparticles, int & numaddback)`

Monte carlo code for the degrader from OPAL classic/5.0/src/Solvers/CollimatorPhysics.cpp on device. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class documentation. TODO: opencl and mic implementations.

3.24.1.6 `int DKSOPAL::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.

3.24.1.7 `int DKSOPAL::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.

3.24.1.8 `int DKSOPAL::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.

3.24.1.9 `int DKSOPAL::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.

3.24.1.10 `int DKSOPAL::callParallelTTrackerKick (void * r_ptr, void * p_ptr, void * ef_ptr, void * bf_ptr, void * dt_ptr, double charge, double mass, int npart, double c, int streamId = -1)`

Integration code from ParallelTTracker from OPAL. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class docs

3.24.1.11 `int DKSOPAL::callParallelTTrackerPush (void * r_ptr, void * p_ptr, int npart, void * dt_ptr, double dt, double c, bool usedt = false, int streamId = -1)`

Integration code from ParallelTTracker from OPAL. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class docs

3.24.1.12 `int DKSOPAL::callParallelTTrackerPush (void * r_ptr, void * p_ptr, void * dt_ptr, int npart, double c, int streamId = -1)`

Integration code from ParallelTTracker from OPAL. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class docs

3.24.1.13 `int DKSOPAL::callParallelTTrackerPushTransform (void * x_ptr, void * p_ptr, void * lastSec_ptr, void * orient_ptr, int npart, int nsec, void * dt_ptr, double dt, double c, bool usedt = false, int streamId = -1)`

Integration code from ParallelTTracker from OPAL. For specifics check OPAL docs and [CudaCollimatorPhysics](#) class docs

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

- /home/l_locans/gitwork/DKS-src/src/DKSOPAL.h
- /home/l_locans/gitwork/DKS-src/src/DKSOPAL.cpp

3.25 DKSSearchStates Class Reference

Public Member Functions

- [DKSSearchStates](#) (Parameters params)
- void [setCurrentState](#) (Parameters current_state)
- void [setCurrentState](#) (States current_state)
- void [initCurrentState](#) ()
- States [getCurrentState](#) ()
- States [getNextNeighbour](#) ()
- States [getRandomNeighbour](#) ()
- void [getNeighbours](#) (int dist=1)
- bool [nextNeighbourExists](#) ()
- void [moveToNeighbour](#) ()
- void [saveCurrentState](#) (double current_time)
- void [printCurrentState](#) (double time=0.0)
- void [printNeighbour](#) (double time=0.0)
- void [printInfo](#) ()
- void [printBest](#) ()

3.25.1 Constructor & Destructor Documentation

3.25.1.1 DKSSearchStates::DKSSearchStates (Parameters *params*)

Constructor always takes params array as variable. Params array is needed to know how many params will be searched and what are thou bounds of each parameter.

set the current state so that number of parameters and parameter bounds are known

3.25.2 Member Function Documentation

3.25.2.1 States DKSSearchStates::getCurrentState ()

get current state

3.25.2.2 void DKSSearchStates::getNeighbours (int *dist* = 1)

calculate all the neighbour states

Get all the possible neighbours of the current state

3.25.2.3 States DKSSearchStates::getNextNeighbour ()

get next neighbour state. if there are no next neighbore stay at the curretn neighbour

3.25.2.4 States DKSSearchStates::getRandomNeighbour ()

get random neighbour state

3.25.2.5 void DKSSearchStates::initCurrentState ()

init random current state

3.25.2.6 void DKSSearchStates::moveToNeighbour ()

move to next neighbour. set the current state as the next neighbour, calculate the neighbours of the new current state.

3.25.2.7 bool DKSSearchStates::nextNeighbourExists ()

Check if there are more neighbours to evaluate Return true if more neighbors exist, false if we are at the last neighbour

3.25.2.8 void DKSSearchStates::printBest ()

Print the best saved state

3.25.2.9 void DKSSearchStates::printCurrentState (double *time* = 0.0)

Print current state. cout the current state. Mostly used for debugging purposes

3.25.2.10 void DKSSearchStates::printInfo ()

Print info. Print the whole info about the search: current state, current neighbour, total neighbors

3.25.2.11 void DKSSearchStates::printNeighbour (double *time* = 0.0)

Print current neighbour info

3.25.2.12 void DKSSearchStates::saveCurrentState (double *current_time*)

Save the current state and the evaluation time of the current state. If evaluation time of the current state is better than the evaluation time of the best state, save the current state as best.

3.25.2.13 void DKSSearchStates::setCurrentState (Parameters *current_state*)

Set current state using parameter vector

3.25.2.14 void DKSSearchStates::setCurrentState (States *current_state*)

set current state using the state vector

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

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSSearchStates.h
- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSSearchStates.cpp

3.26 DKStream Class Reference

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

- /home/l/_locans/gitwork/DKS-src/src/DKStream.h

3.27 DKSTimer Class Reference

Public Member Functions

- [DKSTimer](#) ()
- void [init](#) (std::string n)
- void [start](#) ()
- void [stop](#) ()
- void [reset](#) ()
- double [gettime](#) ()
- void [print](#) ()

3.27.1 Constructor & Destructor Documentation

3.27.1.1 DKSTimer::DKSTimer ()

Init [DKSTimer](#) by seting timer to zero

3.27.2 Member Function Documentation

3.27.2.1 double DKSTimer::gettime ()

Return elapsed time in seconds. Return the value of timervalue

3.27.2.2 void DKSTimer::init (std::string *n*)

Init the timer Set the name for timer and clear all values

3.27.2.3 void DKSTimer::print ()

Print timer. Print the elapsed time of the timer

3.27.2.4 void DKSTimer::reset ()

Reset timervalue to zero. Set timervalue, timeStart and timeEnd to zero

3.27.2.5 void DKSTimer::start ()

Start the timer. Get the curret time with gettimeofday and save in timeStart

3.27.2.6 void DKSTimer::stop ()

Stop the timer Get the current time with gettimeofday and save in timeEnd Calculate elapsed time by timeEnd - timeStart and add to timervalue

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

- /home/l_locans/gitwork/DKS-src/src/Utility/DKSTimer.h
- /home/l_locans/gitwork/DKS-src/src/Utility/DKSTimer.cpp

3.28 Double3 Struct Reference

Public Attributes

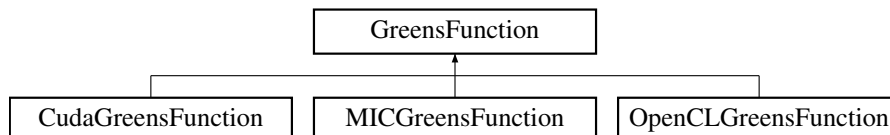
- double **x**
- double **y**
- double **z**

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLCollimatorPhysics.h

3.29 GreensFunction Class Reference

Inheritance diagram for GreensFunction:



Public Member Functions

- virtual int [greensIntegral](#) (void *tmpgreen, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId=-1)=0
- virtual int [integrationGreensFunction](#) (void *rho2_m, void *tmpgreen, int I, int J, int K, int streamId=-1)=0
- virtual int [mirrorRhoField](#) (void *rho2_m, int I, int J, int K, int streamId=-1)=0
- virtual int [multiplyComplexFields](#) (void *ptr1, void *ptr2, int size, int streamId=-1)=0

3.29.1 Member Function Documentation

3.29.1.1 virtual int GreensFunction::greensIntegral (void * *tmpgreen*, int *I*, int *J*, int *K*, int *NI*, int *NJ*, double *hr_m0*, double *hr_m1*, double *hr_m2*, int *streamId* = -1) [pure virtual]

calc greens integral, as defined in OPAL

Implemented in [CudaGreensFunction](#), [OpenCLGreensFunction](#), and [MICGreensFunction](#).

3.29.1.2 `virtual int GreensFunction::integrationGreensFunction (void * rho2_m, void * tmpgreen, int I, int J, int K, int streamId = -1) [pure virtual]`

integration if rho2_m, see OPAL for more details

Implemented in [CudaGreensFunction](#), [OpenCLGreensFunction](#), and [MICGreensFunction](#).

3.29.1.3 `virtual int GreensFunction::mirrorRhoField (void * rho2_m, int I, int J, int K, int streamId = -1) [pure virtual]`

mirror rho2_m field

Implemented in [CudaGreensFunction](#), [OpenCLGreensFunction](#), and [MICGreensFunction](#).

3.29.1.4 `virtual int GreensFunction::multiplyComplexFields (void * ptr1, void * ptr2, int size, int streamId = -1) [pure virtual]`

multiply two complex fields from device memory

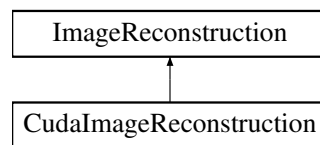
Implemented in [CudaGreensFunction](#), [OpenCLGreensFunction](#), and [MICGreensFunction](#).

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/GreensFunction.h

3.30 ImageReconstruction Class Reference

Inheritance diagram for ImageReconstruction:



Public Member Functions

- `virtual int calculateSource (void *image_space, void *image_position, void *source_position, void *avg, void *std, float diameter, int total_voxels, int total_sources, int start=0)=0`
- `virtual int calculateBackground (void *image_space, void *image_position, void *source_position, void *avg, void *std, float diameter, int total_voxels, int total_sources, int start=0)=0`
- `virtual int calculateSources (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)=0`
- `virtual int calculateBackgrounds (void *image_space, void *image_position, void *source_position, void *avg, void *std, void *diameter, int total_voxels, int total_sources, int start=0)=0`
- `virtual int generateNormalization (void *recon, void *image_position, void *det_position, int total_det)=0`
- `virtual int forwardProjection (void *correction, void *recon, void *list_data, void *det_position, void *image_position, int num_events)=0`
- `virtual int backwardProjection (void *correction, void *recon_corrector, void *list_data, void *det_position, void *image_position, int num_events, int num_voxels)=0`
- `virtual int setDimensions (int voxel_x, int voxel_y, int voxel_z, float voxel_size)=0`
- `virtual int setEdge (float x_edge, float y_edge, float z_edge)=0`
- `virtual int setEdge1 (float x_edge1, float y_edge1, float z_edge1, float z_edge2)=0`
- `virtual int setMinCrystalInRing (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1)=0`
- `virtual int setParams (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter)=0`

Protected Attributes

- void * **m_event_branch**

3.30.1 Member Function Documentation

3.30.1.1 `virtual int ImageReconstruction::backwardProjection (void * correction, void * recon_corrector, void * list_data, void * det_position, void * image_position, int num_events, int num_voxels)` [pure virtual]

Calculate backward projection. For image reconstruction calculates backward projections. see recon.cpp for details
Implemented in [CudaImageReconstruction](#).

3.30.1.2 `virtual int ImageReconstruction::calculateBackground (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0)` [pure virtual]

Calculate background. Places two sphere at each voxel position, calculates the avg value and std value of pixels that are inside the larger sphere, but are outside of the smaller sphere. The diameter of the smaller speher is given by parameter diameter, diameter of the larger sphere is 2*diameter.

Implemented in [CudaImageReconstruction](#).

3.30.1.3 `virtual int ImageReconstruction::calculateBackgrounds (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0)` [pure virtual]

Places two sphere at each voxel position, calculates the avg value and std value of pixels that are inside the larger sphere, but are outside of the smaller sphere. The diameter of the smaller sphere is given by *diameter array, diameter of the larger sphere is 2*diameter of the smaller sphere.

Implemented in [CudaImageReconstruction](#).

3.30.1.4 `virtual int ImageReconstruction::calculateSource (void * image_space, void * image_position, void * source_position, void * avg, void * std, float diameter, int total_voxels, int total_sources, int start = 0)` [pure virtual]

Caluclate source. Places a sphere at each voxel position and calculate the avg value and std value of pixels that are inside this sphere. All the sphere used have the same diameter.

Implemented in [CudaImageReconstruction](#).

3.30.1.5 `virtual int ImageReconstruction::calculateSources (void * image_space, void * image_position, void * source_position, void * avg, void * std, void * diameter, int total_voxels, int total_sources, int start = 0)` [pure virtual]

Caluclate source using differente sources. Places two sphere at each voxel position, calculates the avg value and std value of pixels that are inside the larger sphere, but are outside of the smaller sphere. The diameter of the each sphere is given by *diameter array.

Implemented in [CudaImageReconstruction](#).

3.30.1.6 `virtual int ImageReconstruction::forwardProjection (void * correction, void * recon, void * list_data, void * det_position, void * image_position, int num_events)` [pure virtual]

Calculate forward projection. For image reconstruction calculates forward projections. see recon.cpp for details

Implemented in [CudaImageReconstruction](#).

3.30.1.7 `virtual int ImageReconstruction::generateNormalization (void * recon, void * image_position, void * det_position, int total_det) [pure virtual]`

Generate normalization. Goes through detectors pairs and if detector pair crosses image launches separate kernel that updates voxel values in the image on the slope between these two detectors.

Implemented in [CudaImageReconstruction](#).

3.30.1.8 `virtual int ImageReconstruction::setDimensions (int voxel_x, int voxel_y, int voxel_z, float voxel_size) [pure virtual]`

Set the voxel dimensions on device.

Implemented in [CudaImageReconstruction](#).

3.30.1.9 `virtual int ImageReconstruction::setEdge (float x_edge, float y_edge, float z_edge) [pure virtual]`

Set the image edge variables on the device.

Implemented in [CudaImageReconstruction](#).

3.30.1.10 `virtual int ImageReconstruction::setEdge1 (float x_edge1, float y_edge1, float z_edge1, float z_edge2) [pure virtual]`

Set the image edge1 on the device.

Implemented in [CudaImageReconstruction](#).

3.30.1.11 `virtual int ImageReconstruction::setMinCrystalInRing (float min_CrystalDist_InOneRing, float min_CrystalDist_InOneRing1) [pure virtual]`

Set the minimum crystal in one ring values on the device.

Implemented in [CudaImageReconstruction](#).

3.30.1.12 `virtual int ImageReconstruction::setParams (float matrix_distance_factor, float phantom_diameter, float atten_per_mm, float ring_diameter) [pure virtual]`

Set all other required parameters for reconstruction.

Implemented in [CudaImageReconstruction](#).

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

- `/home/l_locans/gitwork/DKS-src/src/Algorithms/ImageReconstruction.h`

3.31 less_then Struct Reference

Public Member Functions

- `__host__ __device__ bool operator() (int x)`

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

- /home/l_locans/gitwork/DKS-src/src/CUDA/CudaCollimatorPhysics.cu

3.32 ListEvent Struct Reference

Public Attributes

- unsigned **detA**: 16
- unsigned **detB**: 16

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/ImageReconstruction.h

3.33 MICBase Class Reference

Public Member Functions

- int **mic_createRandStreams** (int size)
- int **mic_deleteRandStreams** ()
- int **mic_createStream** (int &streamId)
- int & **mic_getStream** (int id)
- int **mic_deleteStreams** ()
- 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_writeDataAsync** (void *data_ptr, const void *data, int size, int streamId=-1, 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_readDataAsync** (const void *data_ptr, void *result, int size, int streamId=-1, int offset=0)
- int **mic_syncDevice** ()
- 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)

Public Attributes

- void * **defaultRndStream**
- void * **testPtr**
- int **m_device_id**

Protected Attributes

- int **defaultRndSet**

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

- /home/l_locans/gitwork/DKS-src/src/MIC/MICBase.h
- /home/l_locans/gitwork/DKS-src/src/MIC/MICBase.cpp

3.34 MICChiSquare Class Reference

Public Member Functions

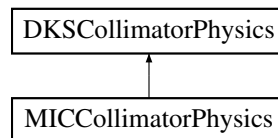
- **MICChiSquare** ([MICBase](#) *base)
- int **mic_chi2** (double *O, double *E, double *result, int size)
- int **mic_Nt** (double *nt, double *p, int psize, int nsize, int jsize, double deltaT=1)
- int **mic_sum** (double *data, double *result, int size)

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

- /home/l_locans/gitwork/DKS-src/src/MIC/MICChiSquare.h
- /home/l_locans/gitwork/DKS-src/src/MIC/MICChiSquare.cpp

3.35 MICCollimatorPhysics Class Reference

Inheritance diagram for MICCollimatorPhysics:



Public Member Functions

- **MICCollimatorPhysics** ([MICBase](#) *base)
- int **CollimatorPhysics** (void *mem_ptr, void *par_ptr, int numparticles, bool enableRutherfordScattering=true)
- int **CollimatorPhysicsSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles)
- int **CollimatorPhysicsSort** (void *mem_ptr, int numparticles, int &numaddback)
- int **CollimatorPhysicsSortSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles, int &numaddback)
- int **ParallelTrackerPush** (void *r_ptr, void *p_ptr, int npart, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)
- int **ParallelTrackerPushTransform** (void *x_ptr, void *p_ptr, void *lastSec_ptr, void *orient_ptr, int npart, int nsec, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)

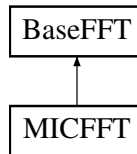
Additional Inherited Members

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

- /home/l_locans/gitwork/DKS-src/src/MIC/MICCollimatorPhysics.h
- /home/l_locans/gitwork/DKS-src/src/MIC/MICCollimatorPhysics.cpp

3.36 MICFFT Class Reference

Inheritance diagram for MICFFT:



Public Member Functions

- **MICFFT** ([MICBase](#) *base)
- int **setupFFT** (int ndim, int N[3])
- int **setupFFTRC** (int ndim, int N[3], double scale=1.0)
- int **setupFFTCTRC** (int ndim, int N[3], double scale=1.0)
- int **executeFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1, bool forward=true)
- int **executeIFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1)
- int **executeRCFFT** (void *in_ptr, void *out_ptr, int ndim, int N[3], int streamId=-1)
- int **executeCRFFT** (void *in_ptr, void *out_ptr, int ndim, int N[3], int streamId=-1)
- int **normalizeFFT** (void *mem_ptr, int ndim, int N[3], int streamId=-1)
- int [destroyFFT](#) ()
- int **normalizeCRFFT** (void *real_ptr, int ndim, int N[3], int streamId=-1)

Additional Inherited Members

3.36.1 Member Function Documentation

3.36.1.1 int MICFFT::destroyFFT () [inline],[virtual]

Info: destroy default FFT plans Return: success or error code

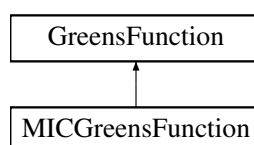
Implements [BaseFFT](#).

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

- /home/l_locans/gitwork/DKS-src/src/MIC/MICFFT.h
- /home/l_locans/gitwork/DKS-src/src/MIC/MICFFT.cpp

3.37 MICGreensFunction Class Reference

Inheritance diagram for MICGreensFunction:



Public Member Functions

- **MICGreensFunction** ([MICBase](#) *base)
- int [greensIntegral](#) (void *tmpgreen_, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId=-1)
- int [integrationGreensFunction](#) (void *rho2_m, void *tmpgreen, int I, int J, int K, int streamId=-1)
- int [mirrorRhoField](#) (void *rho2_m, int I, int J, int K, int streamId=-1)
- int [multiplyCompelxFields](#) (void *ptr1, void *ptr2, int size, int streamId=-1)

3.37.1 Member Function Documentation

3.37.1.1 int [MICGreensFunction::greensIntegral](#) (void * *tmpgreen*, int *I*, int *J*, int *K*, int *NI*, int *NJ*, double *hr_m0*, double *hr_m1*, double *hr_m2*, int *streamId* = -1) [virtual]

calc greens integral, as defined in OPAL

Implements [GreensFunction](#).

3.37.1.2 int [MICGreensFunction::integrationGreensFunction](#) (void * *rho2_m*, void * *tmpgreen*, int *I*, int *J*, int *K*, int *streamId* = -1) [virtual]

integration if rho2_m, see OPAL for more details

Implements [GreensFunction](#).

3.37.1.3 int [MICGreensFunction::mirrorRhoField](#) (void * *rho2_m*, int *I*, int *J*, int *K*, int *streamId* = -1) [virtual]

mirror rho2_m field

Implements [GreensFunction](#).

3.37.1.4 int [MICGreensFunction::multiplyCompelxFields](#) (void * *ptr1*, void * *ptr2*, int *size*, int *streamId* = -1) [virtual]

multiply two complex fields from device memory

Implements [GreensFunction](#).

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

- /home/l_locans/gitwork/DKS-src/src/MIC/MICGreensFunction.hpp
- /home/l_locans/gitwork/DKS-src/src/MIC/MICGreensFunction.cpp

3.38 OpenCLBase Class Reference

Public Member Functions

- int [ocl_createRndStates](#) (int size)
- int [ocl_createRandomNumbers](#) (void *mem_ptr, int size)
- int [ocl_deleteRndStates](#) ()
- int [ocl_getAllDevices](#) ()
- int [ocl_getDeviceCount](#) (int &ndev)
- int [ocl_getDeviceName](#) (std::string &device_name)
- int [ocl_setDevice](#) (int device)
- int [ocl_getUniqueDevices](#) (std::vector< int > &devices)

- int **ocl_setUp** (const char *device_name)
- int **ocl_loadKernel** (const char *kernel_file)
- int **ocl_loadKernelFromSource** (const char *kernel_source, const char *opts=NULL)
- cl_mem **ocl_allocateMemory** (size_t size, int &ierr)
- cl_mem **ocl_allocateMemory** (size_t size, int type, int &ierr)
- template<typename T >
int **ocl_fillMemory** (cl_mem mem_ptr, size_t size, T value, int offset=0)
- 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_group_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)
- int **ocl_checkKernel** (const char *kernel_name, int work_group_size, bool double_precision, int &threads-PerBlock)
- void **ocl_clearEvents** ()
- void **ocl_eventInfo** ()
- cl_command_queue **ocl_getQueue** ()

Static Public Attributes

- static cl_context **m_context** = NULL
- static cl_command_queue **m_command_queue** = NULL

Protected Attributes

- int **defaultRndSet**
- cl_mem **defaultRndState**

3.38.1 Member Function Documentation

3.38.1.1 template<typename T > int OpenCLBase::ocl_fillMemory (cl_mem mem_ptr, size_t size, T value, int offset = 0)
[inline]

Zero OpenCL memory buffer Set all the elements in the device array to zero

3.38.1.2 int OpenCLBase::ocl_getDeviceCount (int & ndev)

Get the OpenCL device count for the set type of device

3.38.1.3 int OpenCLBase::ocl_getDeviceName (std::string & device_name)

Get the name of the device used

3.38.1.4 int OpenCLBase::ocl_getUniqueDevices (std::vector< int > & devices)

Get a list of all the unique devices of the same type that can run OpenCL kernels Used when GPUs of different types might be present on the system.

3.38.1.5 `int OpenCLBase::ocl_loadKernelFromSource (const char * kernel_source, const char * opts = NULL)`

Build program from kernel source. Builds a program from source code provided in `kernel_source`. If compilation fails will return `DKS_ERROR`

3.38.1.6 `int OpenCLBase::ocl_setDevice (int device)`

Set the device to use for OpenCL kernels. device id to use is passed as integer.

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

- `/home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLBase.h`
- `/home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLBase.cpp`

3.39 OpenCLChiSquare Class Reference

Public Member Functions

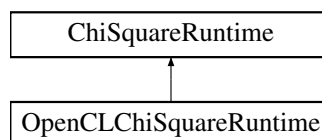
- **OpenCLChiSquare** ([OpenCLBase](#) *base)
- **ocl_PHistoTFFcn** (void *mem_data, void *mem_par, void *mem_result, double fTimeResolution, double fRebin, int sensors, int length, int numpar, double &result)
- **ocl_singleGaussTF** (void *mem_data, void *mem_t0, void *mem_par, void *mem_result, double fTimeResolution, double fRebin, double fGoodBinOffset, int sensors, int length, int numpar, double &result)
- **ocl_doubleLorentzTF** (void *mem_data, void *mem_t0, void *mem_par, void *mem_result, double fTimeResolution, double fRebin, double fGoodBinOffset, int sensors, int length, int numpar, double &result)

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

- `/home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLChiSquare.h`
- `/home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLChiSquare.cpp`

3.40 OpenCLChiSquareRuntime Class Reference

Inheritance diagram for OpenCLChiSquareRuntime:



Public Member Functions

- **OpenCLChiSquareRuntime** ([OpenCLBase](#) *base)
- **OpenCLChiSquareRuntime** ()
- **~OpenCLChiSquareRuntime** ()
- **compileProgram** (std::string function, bool mlh=false)
- **launchChiSquare** (int fitType, void *mem_data, void *mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double &result)
- **writeParams** (const double *params, int numparams)
- **writeFunc** (const double *func, int numfunc)
- **writeMap** (const int *map, int nummap)

- int [initChiSquare](#) (int size_data, int size_param, int size_func, int size_map)
- int [freeChiSquare](#) ()
- int [checkChiSquareKernels](#) (int fitType, int &threadsPerBlock)

Additional Inherited Members

3.40.1 Constructor & Destructor Documentation

3.40.1.1 OpenCLChiSquareRuntime::OpenCLChiSquareRuntime (OpenCLBase * base)

Constructor wiht openclbase argument

3.40.1.2 OpenCLChiSquareRuntime::OpenCLChiSquareRuntime ()

Default constructor

3.40.1.3 OpenCLChiSquareRuntime::~OpenCLChiSquareRuntime ()

Default destructor

3.40.2 Member Function Documentation

3.40.2.1 int OpenCLChiSquareRuntime::checkChiSquareKernels (int fitType, int & threadsPerBlock) [virtual]

Check MuSR kernels for necessary resources. Query device properties to get if sufficient resources are available to run the kernels

Implements [ChiSquareRuntime](#).

3.40.2.2 int OpenCLChiSquareRuntime::compileProgram (std::string function, bool mlh = false) [virtual]

Compile program and save ptx. Add function string to the calcFunction kernel and compile the program Function must be valid C math expression. Parameters can be addressed in a form par[map[idx]]

Implements [ChiSquareRuntime](#).

3.40.2.3 int OpenCLChiSquareRuntime::freeChiSquare () [virtual]

Free temporary memory allocated for chi square. Frees the chisq temporary memory and memory for params, functions and maps

Implements [ChiSquareRuntime](#).

3.40.2.4 int OpenCLChiSquareRuntime::initChiSquare (int size_data, int size_param, int size_func, int size_map) [virtual]

Allocate temporary memory needed for chi square. Initializes the necessary temporary memory for the chi square calculations. Size_data needs to the maximum number of elements in any datasets that will be used for calculations. Size_param, size_func and size_map are the maximum number of parameters, functions and maps used in calculations.

Implements [ChiSquareRuntime](#).

3.40.2.5 `int OpenCLChiSquareRuntime::launchChiSquare (int fitType, void * mem_data, void * mem_err, int length, int numpar, int numfunc, int nummap, double timeStart, double timeStep, double & result)` [virtual]

Launch selected kernel Launched the selected kernel from the compiled code. Result is put in &result variable

Implements [ChiSquareRuntime](#).

3.40.2.6 `int OpenCLChiSquareRuntime::writeFunc (const double * func, int numfunc)` [virtual]

Write functions to device. Write function values from double array to mem_func_m memory on the device.

Implements [ChiSquareRuntime](#).

3.40.2.7 `int OpenCLChiSquareRuntime::writeMap (const int * map, int nummap)` [virtual]

Write maps to device. Write map values from int array to mem_map_m memory on the device.

Implements [ChiSquareRuntime](#).

3.40.2.8 `int OpenCLChiSquareRuntime::writeParams (const double * params, int numparams)` [virtual]

Write params to device. Write params from double array to mem_param_m memory on the device.

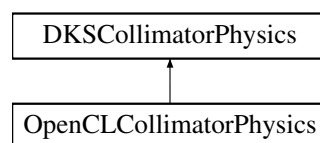
Implements [ChiSquareRuntime](#).

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

- /home/l/_locans/gitwork/DKS-src/src/OpenCL/OpenCLChiSquareRuntime.h
- /home/l/_locans/gitwork/DKS-src/src/OpenCL/OpenCLChiSquareRuntime.cpp

3.41 OpenCLCollimatorPhysics Class Reference

Inheritance diagram for OpenCLCollimatorPhysics:



Public Member Functions

- **OpenCLCollimatorPhysics** ([OpenCLBase](#) *base)
- **CollimatorPhysics** (void *mem_ptr, void *par_ptr, int numparticles, bool enableRutherfordScattering=true)
- **CollimatorPhysicsSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles)
- **CollimatorPhysicsSort** (void *mem_ptr, int numparticles, int &numaddback)
- **CollimatorPhysicsSortSoA** (void *label_ptr, void *localID_ptr, void *rx_ptr, void *ry_ptr, void *rz_ptr, void *px_ptr, void *py_ptr, void *pz_ptr, void *par_ptr, int numparticles, int &numaddback)
- **ParallelTrackerPush** (void *r_ptr, void *p_ptr, int npart, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)
- **ParallelTrackerPushTransform** (void *x_ptr, void *p_ptr, void *lastSec_ptr, void *orient_ptr, int npart, int nsec, void *dt_ptr, double dt, double c, bool usedt=false, int streamId=-1)

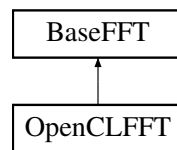
Additional Inherited Members

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLCollimatorPhysics.h
- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLCollimatorPhysics.cpp

3.42 OpenCLFFT Class Reference

Inheritance diagram for OpenCLFFT:



Public Member Functions

- **OpenCLFFT** ([OpenCLBase](#) *base)
- int **executeFFT** (void *data, int ndim, int N[3], int streamId=-1, bool forward=true)
- int **executeIFFT** (void *data, int ndim, int N[3], int streamId=-1)
- int **normalizeFFT** (void *data, int ndim, int N[3], int streamId=-1)
- int **setupFFT** (int ndim, int N[3])
- int **setupFFTRC** (int ndim, int N[3], double scale=1.0)
- int **setupFFTCR** (int ndim, int N[3], double scale=1.0)
- int **destroyFFT** ()
- int **executeRCFFT** (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)
- int **executeCRFFT** (void *real_ptr, void *comp_ptr, int ndim, int N[3], int streamId=-1)
- int **normalizeCRFFT** (void *real_ptr, int ndim, int N[3], int streamId=-1)

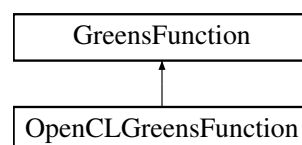
Additional Inherited Members

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLFFT.h
- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLFFT.cpp

3.43 OpenCLGreensFunction Class Reference

Inheritance diagram for OpenCLGreensFunction:



Public Member Functions

- [OpenCLGreensFunction](#) ([OpenCLBase](#) *base)
- [OpenCLGreensFunction](#) ()
- [~OpenCLGreensFunction](#) ()
- [buildProgram](#) ()
- [greensIntegral](#) (void *tmpgreen, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId=-1)
- [integrationGreensFunction](#) (void *rho2_m, void *tmpgreen, int I, int J, int K, int streamId=-1)
- [mirrorRhoField](#) (void *rho2_m, int I, int J, int K, int streamId=-1)
- [multiplyCompelxFields](#) (void *ptr1, void *ptr2, int size, int streamId=-1)

3.43.1 Constructor & Destructor Documentation

3.43.1.1 [OpenCLGreensFunction::OpenCLGreensFunction](#) ([OpenCLBase](#) * base)

Constructor with [OpenCLBase](#) argument

3.43.1.2 [OpenCLGreensFunction::OpenCLGreensFunction](#) ()

Default constructor

3.43.1.3 [OpenCLGreensFunction::~~OpenCLGreensFunction](#) ()

Destructor

3.43.2 Member Function Documentation

3.43.2.1 [int OpenCLGreensFunction::buildProgram](#) ()

Load OpenCL kernel file containing greens function kernels. m_base takes the kernel file and compiles the OpenCL program.

3.43.2.2 [int OpenCLGreensFunction::greensIntegral](#) (void * tmpgreen, int I, int J, int K, int NI, int NJ, double hr_m0, double hr_m1, double hr_m2, int streamId = -1) [virtual]

Info: calc itegral on device memory (taken from OPAL src code) Return: success or error code

Implements [GreensFunction](#).

3.43.2.3 [int OpenCLGreensFunction::integrationGreensFunction](#) (void * rho2_m, void * tmpgreen, int I, int J, int K, int streamId = -1) [virtual]

Info: integration of rho2_m field (taken from OPAL src code) Return: success or error code

Implements [GreensFunction](#).

3.43.2.4 [int OpenCLGreensFunction::mirrorRhoField](#) (void * rho2_m, int I, int J, int K, int streamId = -1) [virtual]

Info: mirror rho field (taken from OPAL src code) Return: succes or error code

Implements [GreensFunction](#).

3.43.2.5 `int OpenCLGreensFunction::multiplyComplexFields (void * ptr1, void * ptr2, int size, int streamId = -1)`
`[virtual]`

Info: multiply complex fields already on the GPU memory, result will be put in ptr1 Return: success or error code

Implements [GreensFunction](#).

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLGreensFunction.h
- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLGreensFunction.cpp

3.44 Parameter Class Reference

Public Member Functions

- **Parameter** (int *_value, int _min, int _max, int _step, std::string _name)
- **Parameter** (double *_value, double _min, double _max, double _step, std::string _name)
- template<typename T >
void **setValue** (T v)
- double **getValue** ()

Public Attributes

- double **min**
- double **max**
- double **step**
- std::string **name**

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

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSSearchStates.h

3.45 PART Struct Reference

Public Attributes

- int **label**
- unsigned **localID**
- double **Rincol** [3]
- double **Pincol** [3]
- long **IDincol**
- int **Binincol**
- double **DTincol**
- double **Qincol**
- long **LastSecincol**
- double **Bfincol** [3]
- double **Efincol** [3]
- int * **label**
- unsigned * **localID**
- double * **rx**
- double * **ry**
- double * **rz**

- double * **px**
- double * **py**
- double * **pz**

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

- /home/l_locans/gitwork/DKS-src/auto-tuning/testCollimatorPhysics.cpp
- /home/l_locans/gitwork/DKS-src/test/testCollimatorPhysicsMPI.cpp
- /home/l_locans/gitwork/DKS-src/test/testCollimatorPhysicsSoA.cpp

3.46 Part Struct Reference

Public Attributes

- double **x**
- double **y**
- double **z**

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

- /home/l_locans/gitwork/DKS-src/test/testMICPush.cpp

3.47 PART_OPENCL Struct Reference

Public Attributes

- int **label**
- unsigned **localID**
- [Double3](#) **Rincol**
- [Double3](#) **Pincol**

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLCollimatorPhysics.h

3.48 PART_SMALL Struct Reference

Public Attributes

- int **label**
- unsigned **localID**
- double **Rincol** [3]
- double **Pincol** [3]

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

- /home/l_locans/gitwork/DKS-src/test/testCollimatorPhysics.cpp

3.49 RNDState Struct Reference

Public Attributes

- double **s10**
- double **s11**
- double **s12**
- double **s20**
- double **s21**
- double **s22**
- double **z**
- bool **gen**

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

- /home/l_locans/gitwork/DKS-src/src/OpenCL/OpenCLBase.h

3.50 State Struct Reference

Public Attributes

- double **value**
- double **min**
- double **max**
- double **step**

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

- /home/l_locans/gitwork/DKS-src/src/AutoTuning/DKSSearchStates.h

3.51 Vector Struct Reference

Public Attributes

- double **x**
- double **y**
- double **z**

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

- /home/l_locans/gitwork/DKS-src/test/testCollimatorPhysics.cpp
- /home/l_locans/gitwork/DKS-src/test/testTimeIntegration.cpp

3.52 VoxelPosition Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

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

- /home/l_locans/gitwork/DKS-src/src/Algorithms/ImageReconstruction.h

3.53 voxelPosition Struct Reference

Public Attributes

- float **x**
- float **y**
- float **z**

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

- /home/l_locans/gitwork/DKS-src/test/testImageReconstruction.cpp

Index

- ~ChiSquareRuntime
 - ChiSquareRuntime, [7](#)
- ~CudaChiSquareRuntime
 - CudaChiSquareRuntime, [15](#)
- ~CudaCollimatorPhysics
 - CudaCollimatorPhysics, [17](#)
- ~CudaFFT
 - CudaFFT, [18](#)
- ~CudaImageReconstruction
 - CudaImageReconstruction, [21](#)
- ~DKSAutoTuning
 - DKSAutoTuning, [23](#)
- ~DKSBase
 - DKSBase, [26](#)
- ~OpenCLChiSquareRuntime
 - OpenCLChiSquareRuntime, [54](#)
- ~OpenCLGreensFunction
 - OpenCLGreensFunction, [57](#)
- addConfigParameter
 - DKSConfig, [34](#)
- addParameter
 - DKSAutoTuning, [23](#)
- allocateHostMemory
 - DKSBase, [26](#)
- allocateMemory
 - DKSBase, [26](#)
- apiCuda
 - DKSBase, [26](#)
- apiOpenCL
 - DKSBase, [26](#)
- apiOpenMP
 - DKSBase, [27](#)
- backwardProjection
 - CudaImageReconstruction, [21](#)
 - ImageReconstruction, [46](#)
- BaseFFT, [5](#)
- buildProgram
 - OpenCLGreensFunction, [57](#)
- CUDA_PART2, [8](#)
- CUDA_PART2_SMALL, [8](#)
- CUDA_PART_SMALL, [9](#)
- calculateBackground
 - CudaImageReconstruction, [21](#)
 - ImageReconstruction, [46](#)
- calculateBackgrounds
 - CudaImageReconstruction, [21](#)
 - ImageReconstruction, [46](#)
- calculateSource
 - CudaImageReconstruction, [21](#)
 - ImageReconstruction, [46](#)
- calculateSources
 - CudaImageReconstruction, [21](#)
 - ImageReconstruction, [46](#)
- callAutoTuningChiSquare
 - DKSBaseMuSR, [32](#)
- callBackwardProjection
 - DKSImageRecon, [37](#)
- callC2RFFT
 - DKSFFT, [35](#)
- callCalculateBackground
 - DKSImageRecon, [37](#)
- callCalculateBackgrounds
 - DKSImageRecon, [37](#)
- callCalculateSource
 - DKSImageRecon, [37](#)
- callCalculateSources
 - DKSImageRecon, [37](#)
- callCollimatorPhysics
 - DKSOPAL, [39](#)
- callCollimatorPhysics2
 - DKSOPAL, [39](#)
- callCollimatorPhysicsSoA
 - DKSOPAL, [39](#)
- callCollimatorPhysicsSort
 - DKSOPAL, [39](#)
- callCollimatorPhysicsSortSoA
 - DKSOPAL, [39](#)
- callCompileProgram
 - DKSBaseMuSR, [32](#)
- callCreateRandomNumbers
 - DKSBase, [27](#)
- callFFT
 - DKSFFT, [35](#)
- callForwardProjection
 - DKSImageRecon, [37](#)
- callGenerateNormalization
 - DKSImageRecon, [37](#)
- callGreensIntegral
 - DKSOPAL, [40](#)
- callGreensIntegration
 - DKSOPAL, [40](#)
- callIFFT
 - DKSFFT, [36](#)
- callInitRandoms
 - DKSBase, [27](#)
- callLaunchChiSquare

- DKSBaseMuSR, [32](#)
- callMemInfo
 - DKSBase, [27](#)
- callMirrorRhoField
 - DKSOPAL, [40](#)
- callMultiplyComplexFields
 - DKSOPAL, [40](#)
- callNormalizeC2RFFT
 - DKSFFT, [36](#)
- callNormalizeFFT
 - DKSFFT, [36](#)
- callParallelTTrackerKick
 - DKSOPAL, [40](#)
- callParallelTTrackerPush
 - DKSOPAL, [40](#)
- callParallelTTrackerPushTransform
 - DKSOPAL, [40](#)
- callR2CFFT
 - DKSFFT, [36](#)
- callSetConsts
 - DKSBaseMuSR, [32](#)
- checkChiSquareKernels
 - CudaChiSquareRuntime, [15](#)
 - OpenCLChiSquareRuntime, [54](#)
- checkMuSRKernels
 - DKSBaseMuSR, [32](#)
- ChiSquareRuntime, [6](#)
 - ~ChiSquareRuntime, [7](#)
 - getOperations, [7](#)
 - setConsts, [7](#)
 - setKernelParams, [7](#)
- clearParameters
 - DKSAutoTuning, [23](#)
- closeHandle
 - DKSBase, [27](#)
- CollimatorPhysics
 - CudaCollimatorPhysics, [17](#)
- CollimatorPhysicsSort
 - CudaCollimatorPhysics, [17](#)
- compare_particle, [7](#)
- compare_particle_small, [8](#)
- compileProgram
 - CudaChiSquareRuntime, [15](#)
 - OpenCLChiSquareRuntime, [54](#)
- createStream
 - DKSBase, [27](#)
- cuda_addStream
 - CudaBase, [10](#)
- cuda_allocateHostMemory
 - CudaBase, [10](#)
- cuda_allocateMemory
 - CudaBase, [10](#)
- cuda_cleanUp
 - CudaBase, [10](#)
- cuda_createCurandStates
 - CudaBase, [10](#)
- cuda_createRandomNumbers
 - CudaBase, [11](#)
- cuda_createStream
 - CudaBase, [11](#)
- cuda_defaultStream
 - CudaBase, [11](#)
- cuda_deleteCurandStates
 - CudaBase, [11](#)
- cuda_deleteStream
 - CudaBase, [11](#)
- cuda_deleteStreams
 - CudaBase, [11](#)
- cuda_executeFunction
 - CudaBase, [11](#)
- cuda_freeHostMemory
 - CudaBase, [11](#)
- cuda_freeMemory
 - CudaBase, [11](#)
- cuda_getCublas
 - CudaBase, [11](#)
- cuda_getCurandStates
 - CudaBase, [11](#)
- cuda_getDeviceCount
 - CudaBase, [11](#)
- cuda_getDeviceName
 - CudaBase, [12](#)
- cuda_getDevices
 - CudaBase, [12](#)
- cuda_getStream
 - CudaBase, [12](#)
- cuda_getStreamId
 - CudaBase, [12](#)
- cuda_getUniqueDevices
 - CudaBase, [12](#)
- cuda_hostRegister
 - CudaBase, [12](#)
- cuda_hostUnregister
 - CudaBase, [12](#)
- cuda_memInfo
 - CudaBase, [12](#)
- cuda_numberOfStreams
 - CudaBase, [12](#)
- cuda_pullData
 - CudaBase, [12](#)
- cuda_pushData
 - CudaBase, [12](#)
- cuda_readData
 - CudaBase, [12](#)
- cuda_readDataAsync
 - CudaBase, [13](#)
- cuda_setDevice
 - CudaBase, [13](#)
- cuda_setStream
 - CudaBase, [13](#)
- cuda_setUp
 - CudaBase, [13](#)
- cuda_syncDevice
 - CudaBase, [13](#)
- cuda_writeData
 - CudaBase, [13](#)

- cuda_writeDataAsync
 - CudaBase, 13
- cuda_zeroMemory
 - CudaBase, 13
- cuda_zeroMemoryAsync
 - CudaBase, 13
- CudaBase, 9
 - cuda_addStream, 10
 - cuda_allocateHostMemory, 10
 - cuda_allocateMemory, 10
 - cuda_cleanUp, 10
 - cuda_createCurandStates, 10
 - cuda_createRandomNumbers, 11
 - cuda_createStream, 11
 - cuda_defaultStream, 11
 - cuda_deleteCurandStates, 11
 - cuda_deleteStream, 11
 - cuda_deleteStreams, 11
 - cuda_executeFunction, 11
 - cuda_freeHostMemory, 11
 - cuda_freeMemory, 11
 - cuda_getCublas, 11
 - cuda_getCurandStates, 11
 - cuda_getDeviceCount, 11
 - cuda_getDeviceName, 12
 - cuda_getDevices, 12
 - cuda_getStream, 12
 - cuda_getStreamId, 12
 - cuda_getUniqueDevices, 12
 - cuda_hostRegister, 12
 - cuda_hostUnregister, 12
 - cuda_memInfo, 12
 - cuda_numberOfStreams, 12
 - cuda_pullData, 12
 - cuda_pushData, 12
 - cuda_readData, 12
 - cuda_readDataAsync, 13
 - cuda_setDevice, 13
 - cuda_setStream, 13
 - cuda_setUp, 13
 - cuda_syncDevice, 13
 - cuda_writeData, 13
 - cuda_writeDataAsync, 13
 - cuda_zeroMemory, 13
 - cuda_zeroMemoryAsync, 13
- CudaChiSquare, 14
 - CudaChiSquare, 14
 - CudaChiSquare, 14
- CudaChiSquareRuntime, 14
 - ~CudaChiSquareRuntime, 15
 - checkChiSquareKernels, 15
 - compileProgram, 15
 - CudaChiSquareRuntime, 15
 - CudaChiSquareRuntime, 15
 - freeChiSquare, 15
 - initChiSquare, 15
 - launchChiSquare, 15
 - writeFunc, 16
 - writeMap, 16
 - writeParams, 16
- CudaCollimatorPhysics, 16
 - ~CudaCollimatorPhysics, 17
 - CollimatorPhysics, 17
 - CollimatorPhysicsSort, 17
 - CudaCollimatorPhysics, 17
 - CudaCollimatorPhysics, 17
 - ParallelTTrackerPush, 17
 - ParallelTTrackerPushTransform, 17
- CudaFFT, 18
 - ~CudaFFT, 18
 - CudaFFT, 18
 - CudaFFT, 18
 - destroyFFT, 19
 - setupFFT, 19
- CudaGreensFunction, 19
 - CudaGreensFunction, 19
 - CudaGreensFunction, 19
 - greensIntegral, 19
 - integrationGreensFunction, 19
 - mirrorRhoField, 20
 - multiplyComplexFields, 20
- CudalImageReconstruction, 20
 - ~CudalImageReconstruction, 21
 - backwardProjection, 21
 - calculateBackground, 21
 - calculateBackgrounds, 21
 - calculateSource, 21
 - calculateSources, 21
 - CudalImageReconstruction, 21
 - CudalImageReconstruction, 21
 - forwardProjection, 22
 - generateNormalization, 22
 - setDimensions, 22
 - setEdge, 22
 - setEdge1, 22
 - setMinCrystallnRing, 22
 - setParams, 22
- DKSAutoTuning, 23
 - ~DKSAutoTuning, 23
 - addParameter, 23
 - clearParameters, 23
 - DKSAutoTuning, 23
 - DKSAutoTuning, 23
 - exhaustiveSearch, 23
 - hillClimbing, 23
 - lineSearch, 24
 - setFunction, 24
 - simulatedAnnealing, 24
- DKSAutoTuningTester, 24
- DKSBase, 24
 - ~DKSBase, 26
 - allocateHostMemory, 26
 - allocateMemory, 26
 - apiCuda, 26
 - apiOpenCL, 26
 - apiOpenMP, 27

- callCreateRandomNumbers, [27](#)
- callInitRandoms, [27](#)
- callMemInfo, [27](#)
- closeHandle, [27](#)
- createStream, [27](#)
- DKSBase, [26](#)
- deviceCPU, [27](#)
- deviceGPU, [27](#)
- deviceMIC, [27](#)
- DKSBase, [26](#)
- freeHostMemory, [27](#)
- freeMemory, [28](#)
- getDeviceCount, [28](#)
- getDeviceList, [28](#)
- getDeviceName, [28](#)
- getDevices, [28](#)
- initDevice, [28](#)
- isAutoTuningOn, [28](#)
- isUseConfigOn, [28](#)
- loadOpenCLKernel, [28](#)
- oclClearEvents, [28](#)
- oclEventInfo, [29](#)
- pullData, [29](#)
- pushData, [29](#)
- readData, [29](#)
- readDataAsync, [29](#)
- registerHostMemory, [30](#)
- setAPI, [30](#)
- setAutoTuningOff, [30](#)
- setAutoTuningOn, [30](#)
- setDefaultDevice, [30](#)
- setDevice, [30](#)
- setUseConfigOff, [30](#)
- setUseConfigOn, [30](#)
- setupDevice, [30](#)
- syncDevice, [30](#)
- unregisterHostMemory, [30](#)
- writeData, [31](#)
- writeDataAsync, [31](#)
- DKSBaseMuSR, [31](#)
 - callAutoTuningChiSquare, [32](#)
 - callCompileProgram, [32](#)
 - callLaunchChiSquare, [32](#)
 - callSetConsts, [32](#)
 - checkMuSRKernels, [32](#)
 - freeChiSquare, [33](#)
 - getOperations, [33](#)
 - initChiSquare, [33](#)
 - testAutoTuning, [33](#)
 - writeFunctions, [33](#)
 - writeMaps, [33](#)
 - writeParams, [33](#)
- DKSCollimatorPhysics, [33](#)
- DKSConfig, [34](#)
 - addConfigParameter, [34](#)
 - DKSConfig, [34](#)
 - DKSConfig, [34](#)
 - getConfigParameter, [34](#)
 - loadConfigFile, [35](#)
 - saveConfigFile, [35](#)
- DKSFFT, [35](#)
 - callC2RFFT, [35](#)
 - callFFT, [35](#)
 - callIFFT, [36](#)
 - callNormalizeC2RFFT, [36](#)
 - callNormalizeFFT, [36](#)
 - callR2CFFT, [36](#)
 - setupFFT, [36](#)
- DKSImageRecon, [36](#)
 - callBackwardProjection, [37](#)
 - callCalculateBackground, [37](#)
 - callCalculateBackgrounds, [37](#)
 - callCalculateSource, [37](#)
 - callCalculateSources, [37](#)
 - callForwardProjection, [37](#)
 - callGenerateNormalization, [37](#)
 - setDimensions, [38](#)
 - setEdge, [38](#)
 - setEdge1, [38](#)
 - setMinCrystallnRing, [38](#)
 - setParams, [38](#)
- DKSOPAL, [38](#)
 - callCollimatorPhysics, [39](#)
 - callCollimatorPhysics2, [39](#)
 - callCollimatorPhysicsSoA, [39](#)
 - callCollimatorPhysicsSort, [39](#)
 - callCollimatorPhysicsSortSoA, [39](#)
 - callGreensIntegral, [40](#)
 - callGreensIntegration, [40](#)
 - callMirrorRhoField, [40](#)
 - callMultiplyComplexFields, [40](#)
 - callParallelTTrackerKick, [40](#)
 - callParallelTTrackerPush, [40](#)
 - callParallelTTrackerPushTransform, [40](#)
- DKSSearchStates, [41](#)
 - DKSSearchStates, [41](#)
 - DKSSearchStates, [41](#)
 - getCurrentState, [41](#)
 - getNeighbours, [41](#)
 - getNextNeighbour, [41](#)
 - getRandomNeighbour, [41](#)
 - initCurrentState, [41](#)
 - moveToNeighbour, [42](#)
 - nextNeighbourExists, [42](#)
 - printBest, [42](#)
 - printCurrentState, [42](#)
 - printInfo, [42](#)
 - printNeighbour, [42](#)
 - saveCurrentState, [42](#)
 - setCurrentState, [42](#)
- DKSStream, [43](#)
- DKSTimer, [43](#)
 - DKSTimer, [43](#)
 - DKSTimer, [43](#)
 - gettime, [43](#)
 - init, [43](#)

- print, [43](#)
- reset, [43](#)
- start, [43](#)
- stop, [43](#)
- destroyFFT
 - CudaFFT, [19](#)
 - MICFFT, [50](#)
- Device, [23](#)
- deviceCPU
 - DKSBase, [27](#)
- deviceGPU
 - DKSBase, [27](#)
- deviceMIC
 - DKSBase, [27](#)
- Double3, [44](#)
- exhaustiveSearch
 - DKSAutoTuning, [23](#)
- forwardProjection
 - CudaImageReconstruction, [22](#)
 - ImageReconstruction, [46](#)
- freeChiSquare
 - CudaChiSquareRuntime, [15](#)
 - DKSBaseMuSR, [33](#)
 - OpenCLChiSquareRuntime, [54](#)
- freeHostMemory
 - DKSBase, [27](#)
- freeMemory
 - DKSBase, [28](#)
- generateNormalization
 - CudaImageReconstruction, [22](#)
 - ImageReconstruction, [47](#)
- getConfigParameter
 - DKSConfig, [34](#)
- getCurrentState
 - DKSSearchStates, [41](#)
- getDeviceCount
 - DKSBase, [28](#)
- getDeviceList
 - DKSBase, [28](#)
- getDeviceName
 - DKSBase, [28](#)
- getDevices
 - DKSBase, [28](#)
- getNeighbours
 - DKSSearchStates, [41](#)
- getNextNeighbour
 - DKSSearchStates, [41](#)
- getOperations
 - ChiSquareRuntime, [7](#)
 - DKSBaseMuSR, [33](#)
- getRandomNeighbour
 - DKSSearchStates, [41](#)
- gettime
 - DKSTimer, [43](#)
- GreensFunction, [44](#)
 - greensIntegral, [44](#)
 - integrationGreensFunction, [44](#)
 - mirrorRhoField, [45](#)
 - multiplyComplexFields, [45](#)
- greensIntegral
 - CudaGreensFunction, [19](#)
 - GreensFunction, [44](#)
 - MICGreensFunction, [51](#)
 - OpenCLGreensFunction, [57](#)
- hillClimbing
 - DKSAutoTuning, [23](#)
- ImageReconstruction, [45](#)
 - backwardProjection, [46](#)
 - calculateBackground, [46](#)
 - calculateBackgrounds, [46](#)
 - calculateSource, [46](#)
 - calculateSources, [46](#)
 - forwardProjection, [46](#)
 - generateNormalization, [47](#)
 - setDimensions, [47](#)
 - setEdge, [47](#)
 - setEdge1, [47](#)
 - setMinCrystalInRing, [47](#)
 - setParams, [47](#)
- init
 - DKSTimer, [43](#)
- initChiSquare
 - CudaChiSquareRuntime, [15](#)
 - DKSBaseMuSR, [33](#)
 - OpenCLChiSquareRuntime, [54](#)
- initCurrentState
 - DKSSearchStates, [41](#)
- initDevice
 - DKSBase, [28](#)
- integrationGreensFunction
 - CudaGreensFunction, [19](#)
 - GreensFunction, [44](#)
 - MICGreensFunction, [51](#)
 - OpenCLGreensFunction, [57](#)
- isAutoTuningOn
 - DKSBase, [28](#)
- isUseConfigOn
 - DKSBase, [28](#)
- launchChiSquare
 - CudaChiSquareRuntime, [15](#)
 - OpenCLChiSquareRuntime, [54](#)
- less_than, [47](#)
- lineSearch
 - DKSAutoTuning, [24](#)
- ListEvent, [48](#)
- loadConfigFile
 - DKSConfig, [35](#)
- loadOpenCLKernel
 - DKSBase, [28](#)
- MICBase, [48](#)
- MICChiSquare, [49](#)

MICCollimatorPhysics, 49
 MICFFT, 50
 destroyFFT, 50
 MIGreensFunction, 50
 greensIntegral, 51
 integrationGreensFunction, 51
 mirrorRhoField, 51
 multiplyCompelxFields, 51
 mirrorRhoField
 CudaGreensFunction, 20
 GreensFunction, 45
 MIGreensFunction, 51
 OpenCLGreensFunction, 57
 moveToNeighbour
 DKSSearchStates, 42
 multiplyCompelxFields
 CudaGreensFunction, 20
 GreensFunction, 45
 MIGreensFunction, 51
 OpenCLGreensFunction, 57

 nextNeighbourExists
 DKSSearchStates, 42

 ocl_fillMemory
 OpenCLBase, 52
 ocl_getDeviceCount
 OpenCLBase, 52
 ocl_getDeviceName
 OpenCLBase, 52
 ocl_getUniqueDevices
 OpenCLBase, 52
 ocl_loadKernelFromSource
 OpenCLBase, 52
 ocl_setDevice
 OpenCLBase, 53
 oclClearEvents
 DKSBase, 28
 oclEventInfo
 DKSBase, 29
 OpenCLBase, 51
 ocl_fillMemory, 52
 ocl_getDeviceCount, 52
 ocl_getDeviceName, 52
 ocl_getUniqueDevices, 52
 ocl_loadKernelFromSource, 52
 ocl_setDevice, 53
 OpenCLChiSquare, 53
 OpenCLChiSquareRuntime, 53
 ~OpenCLChiSquareRuntime, 54
 checkChiSquareKernels, 54
 compileProgram, 54
 freeChiSquare, 54
 initChiSquare, 54
 launchChiSquare, 54
 OpenCLChiSquareRuntime, 54
 OpenCLChiSquareRuntime, 54
 writeFunc, 55
 writeMap, 55
 writeParams, 55
 OpenCLCollimatorPhysics, 55
 OpenCLFFT, 56
 OpenCLGreensFunction, 56
 ~OpenCLGreensFunction, 57
 buildProgram, 57
 greensIntegral, 57
 integrationGreensFunction, 57
 mirrorRhoField, 57
 multiplyCompelxFields, 57
 OpenCLGreensFunction, 57
 OpenCLGreensFunction, 57

 PART, 58
 PART_OPENCL, 59
 PART_SMALL, 59
 ParallelTrackerPush
 CudaCollimatorPhysics, 17
 ParallelTrackerPushTransform
 CudaCollimatorPhysics, 17
 Parameter, 58
 Part, 59
 print
 DKSTimer, 43
 printBest
 DKSSearchStates, 42
 printCurrentState
 DKSSearchStates, 42
 printInfo
 DKSSearchStates, 42
 printNeighbour
 DKSSearchStates, 42
 pullData
 DKSBase, 29
 pushData
 DKSBase, 29

 RNDState, 60
 readData
 DKSBase, 29
 readDataAsync
 DKSBase, 29
 registerHostMemory
 DKSBase, 30
 reset
 DKSTimer, 43

 saveConfigFile
 DKSConfig, 35
 saveCurrentState
 DKSSearchStates, 42
 setAPI
 DKSBase, 30
 setAutoTuningOff
 DKSBase, 30
 setAutoTuningOn
 DKSBase, 30
 setConsts
 ChiSquareRuntime, 7

- setCurrentState
 - DKSSearchStates, [42](#)
- setDefaultDevice
 - DKSBase, [30](#)
- setDevice
 - DKSBase, [30](#)
- setDimensions
 - CudalImageReconstruction, [22](#)
 - DKSImageRecon, [38](#)
 - ImageReconstruction, [47](#)
- setEdge
 - CudalImageReconstruction, [22](#)
 - DKSImageRecon, [38](#)
 - ImageReconstruction, [47](#)
- setEdge1
 - CudalImageReconstruction, [22](#)
 - DKSImageRecon, [38](#)
 - ImageReconstruction, [47](#)
- setFunction
 - DKSAutoTuning, [24](#)
- setKernelParams
 - ChiSquareRuntime, [7](#)
- setMinCrystalInRing
 - CudalImageReconstruction, [22](#)
 - DKSImageRecon, [38](#)
 - ImageReconstruction, [47](#)
- setParams
 - CudalImageReconstruction, [22](#)
 - DKSImageRecon, [38](#)
 - ImageReconstruction, [47](#)
- setUseConfigOff
 - DKSBase, [30](#)
- setUseConfigOn
 - DKSBase, [30](#)
- setupDevice
 - DKSBase, [30](#)
- setupFFT
 - CudaFFT, [19](#)
 - DKSFFT, [36](#)
- simulatedAnnealing
 - DKSAutoTuning, [24](#)
- start
 - DKSTimer, [43](#)
- State, [60](#)
- stop
 - DKSTimer, [43](#)
- syncDevice
 - DKSBase, [30](#)
- testAutoTuning
 - DKSBaseMuSR, [33](#)
- unregisterHostMemory
 - DKSBase, [30](#)
- Vector, [60](#)
- VoxelPosition, [60](#)
- voxelPosition, [61](#)
- writeData
 - DKSBase, [31](#)
- writeDataAsync
 - DKSBase, [31](#)
- writeFunc
 - CudaChiSquareRuntime, [16](#)
 - OpenCLChiSquareRuntime, [55](#)
- writeFunctions
 - DKSBaseMuSR, [33](#)
- writeMap
 - CudaChiSquareRuntime, [16](#)
 - OpenCLChiSquareRuntime, [55](#)
- writeMaps
 - DKSBaseMuSR, [33](#)
- writeParams
 - CudaChiSquareRuntime, [16](#)
 - DKSBaseMuSR, [33](#)
 - OpenCLChiSquareRuntime, [55](#)