Developer Reference

Generated by Doxygen 1.6.1 (the GP-you Group, January 2012)

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

Class Documentation

2.1 GPUtypeS::counter Struct Reference

Public Member Functions

- **counter** (void *p=0, unsigned c=1)

Public Attributes

- void * **ptr**
- unsigned **count**

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

- GPUmat.hh

2.2 GPUexception Class Reference

Public Member Functions

- **GPUexception** (const char ∗err)
- char ∗ **getError** ()

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

- GPUmat.hh

2.3 GPUmat Struct Reference

Public Attributes

- GPUmatGPUtype **gputype**
- GPUmatNumerics **numerics**
- GPUmatFFT **fft**
- GPUmatInterface * **gmat**
- GPUmatCompiler **comp**
- struct GPUmatModules **mod**
- GPUmatAux **aux**
- GPUmatDebug **debug**
- GPUmatRAND **rand**

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

- GPUmat.hh

2.4 GPUmatAux Struct Reference

Public Attributes

- void(∗ mxAssign)(const GPUtype &LHS, const GPUtype &RHS, int dir, int nrhs, const mxArray ∗prhs[ ])

  mxAssign

- GPUtype(∗ mxSliceDrv)(const GPUtype &RHS, int nrhs, const mxArray ∗prhs[ ])

  mxSlice

2.4.1 Member Data Documentation

2.4.1.1 void(∗ GPUmatAux::mxAssign)(const GPUtype &LHS, const GPUtype &RHS, int dir, int nrhs, const mxArray ∗prhs[ ])

mxAssign Defined in NUMERICS module. Used for 'assign' function compilation mxAssign is used in assign.cpp. It is a wrapper to the GPUmat native mxAssign function, with additional checks and functionality

2.4.1.2 GPUtype(∗ GPUmatAux::mxSliceDrv)(const GPUtype &RHS, int nrhs, const mxArray ∗prhs[ ])

mxSlice Defined in NUMERICS module. Used for 'slice' function compilation mxSlice is used in slice.cpp. It is a wrapper to the GPUmat native mxSlice function, with additional checks and functionality

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

- GPUmat.hh

2.5 GPUmatCompiler Struct Reference

Public Attributes

- int(∗ getCompileMode)()
  getCompileMode
- void(∗ pushGPUtype)(void ∗)
  pushGPUtype
- void(∗ pushMx)(const mxArray ∗)
  pushMx
- void(∗ createMxContext)(mxArray ∗mx)
  createMxContext
- void(∗ registerInstruction)(char ∗str)
  registerInstruction
- void(∗ abort)(STRINGCONST char ∗str)
  abort
- int(∗ getContextGPUtype)(void ∗p)
  getContextGPUtype
- int(∗ getContextMx)(void ∗p)
  getContextMx
- void(∗ functionStart)(STRINGCONST char ∗)
  functionStart
- void(∗ functionSetParamInt)(int)
  setParamInt
- void(∗ functionSetParamFloat)(float)
  setParamFloat
- void(∗ functionSetParamDouble)(double)
  setParamDouble
- void(∗ functionSetParamGPUtype)(const GPUtype ∗)
  setParamGPUtype
- void(∗ functionSetParamMx)(const mxArray ∗)
  setParamMx
- void(∗ functionSetParamMxMx)(int nrhs, const mxArray ∗[ ])
  setParamMxMx
- void(∗ functionEnd)(void)
  registerFunction

2.5.1 Member Data Documentation

2.5.1.1 void(∗ GPUmatCompiler::abort)(STRINGCONST char ∗str)

abort Aborts compilation and writes the specified string error

Parameters:

← ***str*** Aborts the compilation and writes the specified error message.

2.5.1.2 void(∗ GPUmatCompiler::createMxContext)(mxArray ∗mx)

createMxContext Add an mxArray to the compilation context.

2.5.1.3 int(∗ GPUmatCompiler::getCompileMode)()

getCompileMode Returns 1 if GPUmat is running in compilation mode. If the returned value is 1, then the function should either abort or generate the necessary code for compilation

2.5.1.4 int(∗ GPUmatCompiler::getContextGPUtype)(void ∗p)

getContextGPUtype Returns the context ID of the GPUtype or -1 if it is not in the compilation context

Parameters:

← ***p*** pointer to GPUtype (casted to void ∗)

Returns:

-1 if the GPUtype is not it the compilation context, it's ID otherwise

2.5.1.5 int(∗ GPUmatCompiler::getContextMx)(void ∗p)

getContextMx Returns the context ID of the mxArray or -1 if it is not in the compilation context

Parameters:

← ***p*** pointer to mxArray (casted to void ∗)

Returns:

-1 if the mxArray is not it the compilation context, it's ID otherwise

2.5.1.6 void(∗ GPUmatCompiler::pushGPUtype)(void ∗)

pushGPUtype GPUtype must be pushed into the compilation context to make it available to other functions.

2.5.1.7 void(∗ GPUmatCompiler::pushMx)(const mxArray ∗)

pushMx Push an mxArray to the compiler context (stack).

2.5.1.8 void(* GPUmatCompiler::registerInstruction)(char *str)

registerInstruction

Parameters:

← ***str*** Register the string str in the compilation buffer. Basically it writes to the generated file the string str.

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

- GPUmat.hh

2.6 GPUmatDebug Struct Reference

Public Attributes

- int(∗ getDebugMode)()
  getDebugMode
- void(∗ setDebugMode)(int)
  setDebugMode
- void(∗ debugPushInstructionStack)(int)
  debugPushInstructionStack
- void(∗ log)(STRINGCONST char ∗str, int level)
  log
- void(∗ logPush)()
  logPush
- void(∗ logPop)()
  logPop
- void(∗ reset)()
  reset

2.6.1 Member Data Documentation

2.6.1.1 void(∗ GPUmatDebug::debugPushInstructionStack)(int)

debugPushInstructionStack Pushes the instruction to the instruction stack for debugging.

2.6.1.2 int(∗ GPUmatDebug::getDebugMode)()

getDebugMode Returns the debug mode flag.

2.6.1.3 void(∗ GPUmatDebug::log)(STRINGCONST char ∗str, int level)

log Writes the specified text to the log.

Parameters:

← ***str*** Text string to be logged

2.6.1.4 void(∗ GPUmatDebug::logPop)()

logPop Log indent pop.

2.6.1.5 void(∗ GPUmatDebug::logPush)()

logPush Log indent push.

2.6.1.6 void(∗ GPUmatDebug::setDebugMode)(int)

setDebugMode Set the debug mode flag.

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

- GPUmat.hh

2.7 GPUmatFFT Struct Reference

Public Attributes

- GPUtype(∗ FFT1Drv)(const GPUtype &p)
  FFT1D.
- GPUtype(∗ FFT2Drv)(const GPUtype &p)
  FFT2D.
- GPUtype(∗ FFT3Drv)(const GPUtype &p)
  FFT3D.
- GPUtype(∗ IFFT1Drv)(const GPUtype &p)
  IFFT1D.
- GPUtype(∗ IFFT2Drv)(const GPUtype &p)
  IFFT2D.
- GPUtype(∗ IFFT3Drv)(const GPUtype &p)
  IFFT3D.

2.7.1 Member Data Documentation

2.7.1.1 GPUtype(∗ GPUmatFFT::FFT1Drv)(const GPUtype &p)

FFT1D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

2.7.1.2 GPUtype(∗ GPUmatFFT::FFT2Drv)(const GPUtype &p)

FFT2D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

2.7.1.3 GPUtype(* GPUmatFFT::FFT3Drv)(const GPUtype &p)

FFT3D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

2.7.1.4 GPUtype(* GPUmatFFT::IFFT1Drv)(const GPUtype &p)

IFFT1D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

2.7.1.5 GPUtype(* GPUmatFFT::IFFT2Drv)(const GPUtype &p)

IFFT2D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

2.7.1.6 GPUtype(* GPUmatFFT::IFFT3Drv)(const GPUtype &p)

IFFT3D.

Parameters:

← ***p*** GPUtype input variable.

Returns:

A new GPUtype object

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

- GPUmat.hh

2.8 GPUmatGPUtype Struct Reference

Public Attributes

- gpuTYPE_t(∗ getType)(const GPUtype &p)
  Returns the type (gpuTYPE_t) of a GPUtype object.
- const int ∗(∗ getSize)(const GPUtype &p)
  Returns the dimensions array of a GPUtype object.
- int(∗ getNdims)(const GPUtype &p)
  Returns the number of dimensions of a GPUtype object.
- int(∗ getNumel)(const GPUtype &p)
  Returns the number of elements of a GPUtype object.
- const void ∗(∗ getGPUptr)(const GPUtype &p)
  Returns the pointer to the GPU memory.
- int(∗ getDataSize)(const GPUtype &p)
  Returns the size of the elements on the GPU memory.
- void(∗ setSize)(const GPUtype &p, int n, const int ∗s)
  Set the size of the GPUtype.
- int(∗ isScalar)(const GPUtype &p)
  Returns 1 if the GPUtype is SCALAR.
- int(∗ isComplex)(const GPUtype &p)
  Returns 1 if the GPUtype is COMPLEX.
- int(∗ isEmpty)(const GPUtype &p)
  Returns 1 if the GPUtype is EMPTY.
- int(∗ isFloat)(const GPUtype &p)
  Returns 1 if the GPUtype is FLOAT (either REAL or COMPLEX).
- int(∗ isDouble)(const GPUtype &p)
  Returns 1 if the GPUtype is DOUBLE (either REAL or COMPLEX).
- GPUtype(∗ create)(gpuTYPE_t type, int ndims, const int ∗size, void ∗init)
  Creates a GPUtype with specified properties: type, number of dimensions, size.
- GPUtype(∗ clone)(const GPUtype &p)
  Clones a GPUtype.
- GPUtype(∗ createMx)(gpuTYPE_t type, int nrhs, const mxArray ∗prhs[ ])
  Creates a GPUtype with specified type.
- mxArray ∗(∗ createMxArray)(const GPUtype &p)

Creates a GPUsingle or GPUdouble object to be returned to Matlab from a given GPUtype.

- mxArray *(* toMxArray)(const GPUtype &p)
  Creates an mxArray from a given GPUtype.
- mxArray *(* createMxArrayPtr)(const mxArray *, const GPUtype &p)
  Internal function.
- GPUtype(* mxToGPUtype)(const mxArray *mx)
  Creates a GPUtype from a Matlab array.
- GPUtype(* getGPUtype)(const mxArray *mx)
  Creates a GPUtype from a Matlab GPUmat variable.
- void(* fill)(const GPUtype &q, double offset, double incr, int m, int p, int offsetp, int type)
  Fills a GPUtype with a sequence of values.
- GPUtype(* colon)(gpuTYPE_t type, double j, double d, double k)
  Similar to the Matlab colon command.
- GPUtype(* slice)(const GPUtype &p, const Range &r)
  Creates a slice from a GPUtype using specified Range.
- GPUtype(* mxSlice)(const GPUtype &p, const Range &r)
  Creates a slice from a GPUtype using specified Range.
- void(* assign)(const GPUtype &p, const GPUtype &q, const Range &r, int dir)
  Assigns a GPUtype to another.
- void(* mxAssign)(const GPUtype &p, const GPUtype &q, const Range &r, int dir)
  Assigns a GPUtype to another.
- void(* permute)(const GPUtype &p, const GPUtype &q, const Range &r, int dir, int *perm)
  Assigns with permuted indexes a GPUtype to another.
- void(* mxPermute)(const GPUtype &p, const GPUtype &q, const Range &r, int dir, int *perm)
  Assigns with permuted indexes a GPUtype to another.
- void(* realimag)(const GPUtype &cpx, const GPUtype &re, const GPUtype &im, int dir, int mode)
  Converts real to complex and complex to real.
- GPUtype(* floatToDouble)(const GPUtype &p)
  Cast from FLOAT to DOUBLE.
- GPUtype(* doubleToFloat)(const GPUtype &p)
  Cast from DOUBLE to FLOAT.
- GPUtype(* realToComplex)(const GPUtype &p)
  Cast from REAL to COMPLEX.

- GPUtype(∗ realImagToComplex)(const GPUtype &re, const GPUtype &im)
  Cast from REAL to COMPLEX.
- GPUtype(∗ mxRepmatDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXREPMAT.
- GPUtype(∗ mxPermuteDrv)(const GPUtype &RHS, int nrhs, const mxArray ∗prhs[ ])
  mxPermuteDrv
- GPUtype(∗ mxEyeDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXEYEDRV.
- GPUtype(∗ mxZerosDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXZEROSDRV.
- GPUtype(∗ mxOnesDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXONESDRV.
- void(∗ eye)(const GPUtype &p)
  EYE.
- void(∗ zeros)(const GPUtype &p)
  ZEROS.
- void(∗ ones)(const GPUtype &p)
  ONES.
- void(∗ mxFill)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXFILL.
- GPUtype(∗ mxColonDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXCOLON.
- void(∗ mxMemCpyDtoD)(const GPUtype &dst, const GPUtype &src, int nrhs, const mxArray ∗prhs[ ])
  MXMEMCPYDTOD.
- void(∗ mxMemCpyHtoD)(const GPUtype &dst, int nrhs, const mxArray ∗prhs[ ])
  MXMEMCPYHTOD.

2.8.1 Member Data Documentation

2.8.1.1 void(∗ GPUmatGPUtype::assign)(const GPUtype &p, const GPUtype &q, const Range &r, int dir)

Assigns a GPUtype to another. Range and dir are used to apply the Range to left or right hand side

Parameters:

← ***p*** GPUtype input variable

← ***q*** GPUtype input variable

← ***r*** Range used for the slice

← ***dir*** If 0 Range is applied to q, if 1 Range is applied to p

Returns:

A new GPUtype object

2.8.1.2 GPUtype(∗ GPUmatGPUtype::clone)(const GPUtype &p)

Clones a GPUtype.

Parameters:

← ***p*** GPUtype to clone.

Returns:

The cloned GPUtype

2.8.1.3 GPUtype(∗ GPUmatGPUtype::colon)(gpuTYPE_t type, double j, double d, double k)

Similar to the Matlab colon command. J:K is the same as [J, J+1, ..., K]. J:K is empty if J > K. J:D:K is the same as [J, J+D, ..., J+m∗D] where m = fix((K-J)/D). J:D:K is empty if D == 0, if D > 0 and J > K, or if D < 0 and J < K.

Parameters:

← ***type*** GPUtype type.

2.8.1.4 GPUtype(∗ GPUmatGPUtype::create)(gpuTYPE_t type, int ndims, const int ∗size, void ∗init)

Creates a GPUtype with specified properties: type, number of dimensions, size. Creates a GPUtype with specified properties: type, number of dimensions, size. If ndims = 0 or size=NULL the GPUtype is an empty GPUtype.

Parameters:

← ***type*** GPUtype type (gpuFLOAT, gpuDOUBLE, ...).

← ***ndims*** Number of dimensions.

← ***size*** Dimensions array.

← ***initialization*** vector. Set to NULL if initialization is not required.

Returns:

The created GPUtype

2.8.1.5 GPUtype(* GPUmatGPUtype::createMx)(gpuTYPE_t type, int nrhs, const mxArray *prhs[ ])

Creates a GPUtype with specified type. Dimensions are constructed from input arguments nrhs and prhs. For example, we have the following expression in Matlab:

A = eye(3,4,5,GPUsingle)

The function eye should create an output GPUtype variable with dimensions (3,4,5). The code to perform such operation is the following:

```
GPUtype IN = gm->gputype.getGPUtype(prhs[nrhs-1]);
gpuTYPE_t tin = gm->gputype.getType(IN);
GPUtype r = gm->gputype.createMx(tin, nrhs-1, prhs);
```

Parameters:

← ***type*** GPUtype type (gpuFLOAT, gpuDOUBLE, ...).

← ***nrhs*** Number of elements of array prhs[].

← ***prhs*** Each element specifies a dimension.

Returns:

The created GPUtype

2.8.1.6 mxArray*(* GPUmatGPUtype::createMxArray)(const GPUtype &p)

Creates a GPUsingle or GPUdouble object to be returned to Matlab from a given GPUtype.

Parameters:

← ***p*** GPUtype input variable.

Returns:

Matlab mxArray pointer (GPUsingle or GPUdouble object)

2.8.1.7 GPUtype(* GPUmatGPUtype::doubleToFloat)(const GPUtype &p)

Cast from DOUBLE to FLOAT.

Parameters:

← ***p*** GPUtype input variable (DOUBLE)

Returns:

A new GPUtype object (FLOAT)

2.8.1.8 void(* GPUmatGPUtype::eye)(const GPUtype &p)

EYE. Defined in NUMERICS module

2.8.1.9 void(∗ GPUmatGPUtype::fill)(const GPUtype &q, double offset, double incr, int m, int p, int offsetp, int type)

Fills a GPUtype with a sequence of values. The element of q in position i will have the following value incr∗(i % m) + offset if (((i+offsetp) % p)==0);

Parameters:

← ***q*** GPUtype.

← ***offset*** (see formula above)

← ***incr*** (see formula above)

← ***m*** (see formula above)

← ***p*** (see formula above)

← ***offsetp*** (see formula above)

← ***type=0*** only real part is modified type=1 only imaginary part is modified type=2 both real and imaginary are modified

2.8.1.10 GPUtype(∗ GPUmatGPUtype::floatToDouble)(const GPUtype &p)

Cast from FLOAT to DOUBLE.

Parameters:

← ***p*** GPUtype input variable (FLOAT)

Returns:

A new GPUtype object (DOUBLE)

2.8.1.11 int(∗ GPUmatGPUtype::getDataSize)(const GPUtype &p)

Returns the size of the elements on the GPU memory.

Parameters:

← ***p*** GPUtype input variable.

Returns:

The size of the elements on GPU memory

2.8.1.12 const void∗(∗ GPUmatGPUtype::getGPUptr)(const GPUtype &p)

Returns the pointer to the GPU memory.

Parameters:

← ***p*** GPUtype input variable.

Returns:

The pointer to the GPU memory

2.8.1.13 GPUtype(* GPUmatGPUtype::getGPUtype)(const mxArray *mx)

Creates a GPUtype from a Matlab GPUmat variable.

Parameters:

← ***mx*** GPUmat variable (GPUsingle, GPUdouble).

Returns:

GPUtype object

2.8.1.14 int(* GPUmatGPUtype::getNdims)(const GPUtype &p)

Returns the number of dimensions of a GPUtype object.

Parameters:

← ***p*** GPUtype input variable.

Returns:

The number of dimensions

2.8.1.15 int(* GPUmatGPUtype::getNumel)(const GPUtype &p)

Returns the number of elements of a GPUtype object.

Parameters:

← ***p*** GPUtype input variable.

Returns:

The number of elements

2.8.1.16 const int*(* GPUmatGPUtype::getSize)(const GPUtype &p)

Returns the dimensions array of a GPUtype object.

Parameters:

← ***p*** GPUtype input variable.

Returns:

The dimensions array

2.8.1.17 gpuTYPE_t(∗ GPUmatGPUtype::getType)(const GPUtype &p)

Returns the type (gpuTYPE_t) of a GPUtype object.

Parameters:

← ***p*** GPUtype input variable.

Returns:

gpuTYPE_t

2.8.1.18 int(∗ GPUmatGPUtype::isComplex)(const GPUtype &p)

Returns 1 if the GPUtype is COMPLEX.

Parameters:

← ***p*** GPUtype input variable.

Returns:

1 if COMPLEX

2.8.1.19 int(∗ GPUmatGPUtype::isDouble)(const GPUtype &p)

Returns 1 if the GPUtype is DOUBLE (either REAL or COMPLEX).

Parameters:

← ***p*** GPUtype input variable.

Returns:

1 if DOUBLE

2.8.1.20 int(∗ GPUmatGPUtype::isEmpty)(const GPUtype &p)

Returns 1 if the GPUtype is EMPTY.

Parameters:

← ***p*** GPUtype input variable.

Returns:

1 if EMPTY

2.8.1.21 int(* GPUmatGPUtype::isFloat)(const GPUtype &p)

Returns 1 if the GPUtype is FLOAT (either REAL or COMPLEX).

Parameters:

← ***p*** GPUtype input variable.

Returns:

1 if FLOAT

2.8.1.22 int(* GPUmatGPUtype::isScalar)(const GPUtype &p)

Returns 1 if the GPUtype is SCALAR.

Parameters:

← ***p*** GPUtype input variable.

Returns:

1 if SCALAR

2.8.1.23 void(* GPUmatGPUtype::mxAssign)(const GPUtype &p, const GPUtype &q, const Range &r, int dir)

Assigns a GPUtype to another. Indexes are considered as in Matlab/Fortran (starting from 1) Range and dir are used to apply the Range to left or right hand side

Parameters:

← ***p*** GPUtype input variable

← ***q*** GPUtype input variable

← ***r*** Range used for the slice

← ***dir*** If 0 Range is applied to q, if 1 Range is applied to p

Returns:

A new GPUtype object

2.8.1.24 GPUtype(* GPUmatGPUtype::mxColonDrv)(const GPUtype &p, int nrhs, const mxArray *prhs[ ])

MXCOLON. Wrapper to the colon function Defined in NUMERICS module

2.8.1.25 GPUtype(* GPUmatGPUtype::mxEyeDrv)(const GPUtype &p, int nrhs, const mxArray *prhs[ ])

MXEYEDRV. Defined in NUMERICS module

2.8.1.26 void(∗ GPUmatGPUtype::mxFill)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])

MXFILL. Wrapper to the fill function Defined in NUMERICS module

2.8.1.27 void(∗ GPUmatGPUtype::mxMemCpyDtoD)(const GPUtype &dst, const GPUtype &src, int nrhs, const mxArray ∗prhs[ ])

MXMEMCPYDTOD. Wrapper to the memCpyDtoD function Defined in NUMERICS module

2.8.1.28 void(∗ GPUmatGPUtype::mxMemCpyHtoD)(const GPUtype &dst, int nrhs, const mxArray ∗prhs[ ])

MXMEMCPYHTOD. Wrapper to the memCpyHtoD function Defined in NUMERICS module

2.8.1.29 GPUtype(∗ GPUmatGPUtype::mxOnesDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])

MXONESDRV. Defined in NUMERICS module

2.8.1.30 void(∗ GPUmatGPUtype::mxPermute)(const GPUtype &p, const GPUtype &q, const Range &r, int dir, int ∗perm)

Assigns with permuted indexes a GPUtype to another. Indexes are considered as in Matlab/Fortran (starting from 1) Range and dir are used to apply the Range to left or right hand side

Parameters:

← ***p*** GPUtype input variable

← ***q*** GPUtype input variable

← ***r*** Range used for the slice

← ***dir*** If 0 Range is applied to q, if 1 Range is applied to p

← ***perm*** Array with permutation indexes

Returns:

A new GPUtype object

2.8.1.31 GPUtype(∗ GPUmatGPUtype::mxPermuteDrv)(const GPUtype &RHS, int nrhs, const mxArray ∗prhs[ ])

mxPermuteDrv Defined in NUMERICS module

2.8.1.32 GPUtype(∗ GPUmatGPUtype::mxRepmatDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])

MXREPMAT. Defined in NUMERICS module

2.8.1.33 GPUtype(* GPUmatGPUtype::mxSlice)(const GPUtype &p, const Range &r)

Creates a slice from a GPUtype using specified Range. Indexes are considered as in Matlab/Fortran (starting from 1)

Parameters:

← ***p*** GPUtype input variable

← ***r*** Range used for the slice

Returns:

A new GPUtype object

2.8.1.34 GPUtype(* GPUmatGPUtype::mxToGPUtype)(const mxArray *mx)

Creates a GPUtype from a Matlab array.

Parameters:

← ***mx*** Matlab array.

Returns:

GPUtype

2.8.1.35 GPUtype(* GPUmatGPUtype::mxZerosDrv)(const GPUtype &p, int nrhs, const mxArray *prhs[ ])

MXZEROSDRV. Defined in NUMERICS module

2.8.1.36 void(* GPUmatGPUtype::ones)(const GPUtype &p)

ONES. Defined in NUMERICS module

2.8.1.37 void(* GPUmatGPUtype::permute)(const GPUtype &p, const GPUtype &q, const Range &r, int dir, int *perm)

Assigns with permuted indexes a GPUtype to another. Range and dir are used to apply the Range to left or right hand side

Parameters:

← ***p*** GPUtype input variable

← ***q*** GPUtype input variable

← ***r*** Range used for the slice

← ***dir*** If 0 Range is applied to q, if 1 Range is applied to p

← ***perm*** Array with permutation indexes

Returns:

A new GPUtype object

2.8.1.38 void(∗ GPUmatGPUtype::realimag)(const GPUtype &cpx, const GPUtype &re, const GPUtype &im, int dir, int mode)

Converts real to complex and complex to real. Depending on dir and mode the following operations are performed: dir 0 - REAL to COMPLEX 1 - COMPLEX to REAL mode 0 - REAL, IMAG 1 - REAL 2 - IMAG The following operations are done depending on the combination dir/mode dir mode operation 0 0 re and im -> cpx 0 1 re -> cpx (imaginary part set to zero) 0 2 im -> cpx (real part set to zero) 1 0 cpx -> re and im 1 1 cpx -> re (im is not considered) 1 2 cpx -> im (re is not considered)

Parameters:

← ***cpx*** GPUtype complex

← ***re*** GPUtype real

← ***im*** GPUtype im

← ***dir*** Defines the type of the operation (real to complex, complex to real)

← ***mode*** Defines which GPUtype re or/and im is used

2.8.1.39 GPUtype(∗ GPUmatGPUtype::realImagToComplex)(const GPUtype &re, const GPUtype &im)

Cast from REAL to COMPLEX.

Parameters:

← ***re*** GPUtype input variable (REAL)

← ***im*** GPUtype input variable (IMAG)

Returns:

A new GPUtype object (COMPLEX)

2.8.1.40 GPUtype(∗ GPUmatGPUtype::realToComplex)(const GPUtype &p)

Cast from REAL to COMPLEX.

Parameters:

← ***p*** GPUtype input variable (REAL)

Returns:

A new GPUtype object (COMPLEX)

2.8.1.41 void(∗ GPUmatGPUtype::setSize)(const GPUtype &p, int n, const int ∗s)

Set the size of the GPUtype.

Parameters:

← ***p*** GPUtype input variable.

← ***n*** The number of elements of the array s

← ***s*** Array with dimensions

2.8.1.42 GPUtype(* GPUmatGPUtype::slice)(const GPUtype &p, const Range &r)

Creates a slice from a GPUtype using specified Range.

Parameters:

← ***p*** GPUtype input variable

← ***r*** Range used for the slice

Returns:

A new GPUtype object

2.8.1.43 mxArray*(* GPUmatGPUtype::toMxArray)(const GPUtype &p)

Creates an mxArray from a given GPUtype. This function is different from createMxArray. It creates a Matlab array with the same type (double, single) of the input GPUtype. createMxArray creates an mxArray of type GPUsingle, GPUdouble or any other GPUmat variable.

Parameters:

← ***p*** GPUtype input variable.

Returns:

Matlab mxArray pointer (different from createMxArray)

2.8.1.44 void(* GPUmatGPUtype::zeros)(const GPUtype &p)

ZEROS. Defined in NUMERICS module

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

- GPUmat.hh

2.9 GPUmatInterface Struct Reference

Public Attributes

- struct GPUmatInterfaceFunction **fun**
- struct GPUmatInterfaceConfig **config**
- struct GPUmatManager **control**

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

- GPUmat.hh

2.10 GPUmatInterfaceConfig Struct Reference

Public Attributes

- void(∗ getMajorMinor)(int ∗major, int ∗minor)
  getMajorMinor
- int(∗ getActiveDeviceNumber)()
  getActiveDeviceNumber

2.10.1 Member Data Documentation

2.10.1.1 int(∗ GPUmatInterfaceConfig::getActiveDeviceNumber)()

getActiveDeviceNumber Returns the GPU device being used

Returns:

The device number being used

2.10.1.2 void(∗ GPUmatInterfaceConfig::getMajorMinor)(int ∗major, int ∗minor)

getMajorMinor Returns the GPUmat major and minor version

Parameters:

→ ***major*** Returns the major number
→ ***minor*** Returns the minor number

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

- GPUmat.hh

2.11 GPUmatInterfaceFunction Struct Reference

Public Attributes

- int(∗ registerFunction)(STRINGCONST char ∗name, void ∗f)
  registerFunction
- void ∗(∗ getFunctionByName)(STRINGCONST char ∗name)
  getFunctionByName
- int(∗ getFunctionNumber)(STRINGCONST char ∗name)
  getFunctionNumber
- void ∗(∗ getFunctionByNumber)(int findex)
  getFunctionByNumber

2.11.1 Member Data Documentation

2.11.1.1 void∗(∗ GPUmatInterfaceFunction::getFunctionByName)(STRINGCONST char ∗name)

getFunctionByName Returns the function pointer

Parameters:

← ***name*** Function name.

Returns:

Function pointer

2.11.1.2 void∗(∗ GPUmatInterfaceFunction::getFunctionByNumber)(int findex)

getFunctionByNumber Returns the function pointer

Parameters:

← ***findex*** Function index (returned using registerFunction or getFunctionNumber).

Returns:

Function pointer

2.11.1.3 int(∗ GPUmatInterfaceFunction::getFunctionNumber)(STRINGCONST char ∗name)

getFunctionNumber Returns the function index

Parameters:

← ***name*** Function name.

Returns:

Function index

2.11.1.4 int(* GPUmatInterfaceFunction::registerFunction)(STRINGCONST char *name, void *f)

registerFunction Register a user defined function

Parameters:

← ***name*** Function name.

← ***f*** Function pointer.

Returns:

The index assigned to the function. Can be used to get the function pointer using getFunctionByNumber

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

- GPUmat.hh

2.12 GPUmatManager Struct Reference

Public Attributes

- void(∗ cacheClean)(void)
  cacheClean

2.12.1 Member Data Documentation

2.12.1.1 void(∗ GPUmatManager::cacheClean)(void)

cacheClean Clean GPU memory cache

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

- GPUmat.hh

2.13 GPUmatModules Struct Reference

Public Attributes

- int gpumat
  native GPUmat functions loaded
- int modules
  main module
- int numerics
  numerics module
- int rand
  rand module

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

- GPUmat.hh

2.14 GPUmatNumerics Struct Reference

Public Attributes

- GPUtype(* AbsDrv)(const GPUtype &p)
  AbsDrv function.
- GPUtype(* AcosDrv)(const GPUtype &p)
  AcosDrv function.
- GPUtype(* AcoshDrv)(const GPUtype &p)
  AcoshDrv function.
- GPUtype(* AndDrv)(const GPUtype &p, const GPUtype &q)
  AndDrv function.
- GPUtype(* AsinDrv)(const GPUtype &p)
  AsinDrv function.
- GPUtype(* AsinhDrv)(const GPUtype &p)
  AsinhDrv function.
- GPUtype(* AtanDrv)(const GPUtype &p)
  AtanDrv function.
- GPUtype(* AtanhDrv)(const GPUtype &p)
  AtanhDrv function.
- GPUtype(* CeilDrv)(const GPUtype &p)
  CeilDrv function.
- GPUtype(* ConjDrv)(const GPUtype &p)
  ConjDrv function.
- GPUtype(* CosDrv)(const GPUtype &p)
  CosDrv function.
- GPUtype(* CoshDrv)(const GPUtype &p)
  CoshDrv function.
- GPUtype(* CtransposeDrv)(const GPUtype &p)
  CtransposeDrv function.
- GPUtype(* EqDrv)(const GPUtype &p, const GPUtype &q)
  EqDrv function.
- GPUtype(* ExpDrv)(const GPUtype &p)
  ExpDrv function.
- GPUtype(* FloorDrv)(const GPUtype &p)

FloorDrv function.

- GPUtype(* GeDrv)(const GPUtype &p, const GPUtype &q)
  GeDrv function.
- void(* Abs)(const GPUtype &p, const GPUtype &q)
  Abs function.
- void(* Acos)(const GPUtype &p, const GPUtype &q)
  Acos function.
- void(* Acosh)(const GPUtype &p, const GPUtype &q)
  Acosh function.
- void(* And)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  And function.
- void(* Asin)(const GPUtype &p, const GPUtype &q)
  Asin function.
- void(* Asinh)(const GPUtype &p, const GPUtype &q)
  Asinh function.
- void(* Atan)(const GPUtype &p, const GPUtype &q)
  Atan function.
- void(* Atanh)(const GPUtype &p, const GPUtype &q)
  Atanh function.
- void(* Ceil)(const GPUtype &p, const GPUtype &q)
  Ceil function.
- void(* Conj)(const GPUtype &p, const GPUtype &q)
  Conj function.
- void(* Cos)(const GPUtype &p, const GPUtype &q)
  Cos function.
- void(* Cosh)(const GPUtype &p, const GPUtype &q)
  Cosh function.
- void(* Ctranspose)(const GPUtype &p, const GPUtype &q)
  Ctranspose function.
- void(* Eq)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Eq function.
- void(* Exp)(const GPUtype &p, const GPUtype &q)
  Exp function.

- void(* Floor)(const GPUtype &p, const GPUtype &q)
  Floor function.
- void(* Ge)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Ge function.
- void(* Gt)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Gt function.
- void(* Imag)(const GPUtype &p, const GPUtype &q)
  Imag function.
- void(* Ldivide)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Ldivide function.
- void(* Le)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Le function.
- void(* Log)(const GPUtype &p, const GPUtype &q)
  Log function.
- void(* Log10)(const GPUtype &p, const GPUtype &q)
  Log10 function.
- void(* Log1p)(const GPUtype &p, const GPUtype &q)
  Log1p function.
- void(* Log2)(const GPUtype &p, const GPUtype &q)
  Log2 function.
- void(* Lt)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Lt function.
- void(* Minus)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Minus function.
- void(* Mtimes)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Mtimes function.
- void(* Ne)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Ne function.
- void(* Not)(const GPUtype &p, const GPUtype &q)
  Not function.
- void(* Or)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Or function.
- void(* Plus)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Plus function.

- void(* Power)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Power function.
- void(* Rdivide)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Rdivide function.
- void(* Real)(const GPUtype &p, const GPUtype &q)
  Real function.
- void(* Round)(const GPUtype &p, const GPUtype &q)
  Round function.
- void(* Sin)(const GPUtype &p, const GPUtype &q)
  Sin function.
- void(* Sinh)(const GPUtype &p, const GPUtype &q)
  Sinh function.
- void(* Sqrt)(const GPUtype &p, const GPUtype &q)
  Sqrt function.
- void(* Tan)(const GPUtype &p, const GPUtype &q)
  Tan function.
- void(* Tanh)(const GPUtype &p, const GPUtype &q)
  Tanh function.
- void(* Times)(const GPUtype &p, const GPUtype &q, const GPUtype &r)
  Times function.
- void(* Transpose)(const GPUtype &p, const GPUtype &q)
  Transpose function.
- void(* Uminus)(const GPUtype &p, const GPUtype &q)
  Uminus function.
- GPUtype(* GtDrv)(const GPUtype &p, const GPUtype &q)
  GtDrv function.
- GPUtype(* ImagDrv)(const GPUtype &p)
  ImagDrv function.
- GPUtype(* LdivideDrv)(const GPUtype &p, const GPUtype &q)
  LdivideDrv function.
- GPUtype(* LeDrv)(const GPUtype &p, const GPUtype &q)
  LeDrv function.
- GPUtype(* LogDrv)(const GPUtype &p)

LogDrv function.

- GPUtype(* Log10Drv)(const GPUtype &p)
  Log10Drv function.
- GPUtype(* Log1pDrv)(const GPUtype &p)
  Log1pDrv function.
- GPUtype(* Log2Drv)(const GPUtype &p)
  Log2Drv function.
- GPUtype(* LtDrv)(const GPUtype &p, const GPUtype &q)
  LtDrv function.
- GPUtype(* MinusDrv)(const GPUtype &p, const GPUtype &q)
  MinusDrv function.
- GPUtype(* MtimesDrv)(const GPUtype &p, const GPUtype &q)
  MtimesDrv function.
- GPUtype(* NeDrv)(const GPUtype &p, const GPUtype &q)
  NeDrv function.
- GPUtype(* NotDrv)(const GPUtype &p)
  NotDrv function.
- GPUtype(* OrDrv)(const GPUtype &p, const GPUtype &q)
  OrDrv function.
- GPUtype(* PlusDrv)(const GPUtype &p, const GPUtype &q)
  PlusDrv function.
- GPUtype(* PowerDrv)(const GPUtype &p, const GPUtype &q)
  PowerDrv function.
- GPUtype(* RdivideDrv)(const GPUtype &p, const GPUtype &q)
  RdivideDrv function.
- GPUtype(* RealDrv)(const GPUtype &p)
  RealDrv function.
- GPUtype(* RoundDrv)(const GPUtype &p)
  RoundDrv function.
- GPUtype(* SinDrv)(const GPUtype &p)
  SinDrv function.
- GPUtype(* SinhDrv)(const GPUtype &p)
  SinhDrv function.

- GPUtype(∗ SqrtDrv)(const GPUtype &p)

  SqrtDrv function.

- GPUtype(∗ TanDrv)(const GPUtype &p)

  TanDrv function.

- GPUtype(∗ TanhDrv)(const GPUtype &p)

  TanhDrv function.

- GPUtype(∗ TimesDrv)(const GPUtype &p, const GPUtype &q)

  TimesDrv function.

- GPUtype(∗ TransposeDrv)(const GPUtype &p)

  TransposeDrv function.

- GPUtype(∗ UminusDrv)(const GPUtype &p)

  UminusDrv function.

2.14.1 Member Data Documentation

2.14.1.1 void(∗ GPUmatNumerics::Abs)(const GPUtype &p, const GPUtype &q)

Abs function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.2 GPUtype(∗ GPUmatNumerics::AbsDrv)(const GPUtype &p)

AbsDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.3 void(∗ GPUmatNumerics::Acos)(const GPUtype &p, const GPUtype &q)

Acos function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.4 GPUtype(∗ GPUmatNumerics::AcosDrv)(const GPUtype &p)

AcosDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.5 void(∗ GPUmatNumerics::Acosh)(const GPUtype &p, const GPUtype &q)

Acosh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.6 GPUtype(∗ GPUmatNumerics::AcoshDrv)(const GPUtype &p)

AcoshDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.7 void(∗ GPUmatNumerics::And)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

And function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.8 GPUtype(∗ GPUmatNumerics::AndDrv)(const GPUtype &p, const GPUtype &q)

AndDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.9 void(∗ GPUmatNumerics::Asin)(const GPUtype &p, const GPUtype &q)

Asin function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.10 GPUtype(∗ GPUmatNumerics::AsinDrv)(const GPUtype &p)

AsinDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.11 void(∗ GPUmatNumerics::Asinh)(const GPUtype &p, const GPUtype &q)

Asinh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.12 GPUtype(∗ GPUmatNumerics::AsinhDrv)(const GPUtype &p)

AsinhDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.13 void(∗ GPUmatNumerics::Atan)(const GPUtype &p, const GPUtype &q)

Atan function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.14 GPUtype(∗ GPUmatNumerics::AtanDrv)(const GPUtype &p)

AtanDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.15 void(* GPUmatNumerics::Atanh)(const GPUtype &p, const GPUtype &q)

Atanh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.16 GPUtype(* GPUmatNumerics::AtanhDrv)(const GPUtype &p)

AtanhDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.17 void(* GPUmatNumerics::Ceil)(const GPUtype &p, const GPUtype &q)

Ceil function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.18 GPUtype(* GPUmatNumerics::CeilDrv)(const GPUtype &p)

CeilDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.19 void(∗ GPUmatNumerics::Conj)(const GPUtype &p, const GPUtype &q)

Conj function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.20 GPUtype(∗ GPUmatNumerics::ConjDrv)(const GPUtype &p)

ConjDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.21 void(∗ GPUmatNumerics::Cos)(const GPUtype &p, const GPUtype &q)

Cos function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.22 GPUtype(∗ GPUmatNumerics::CosDrv)(const GPUtype &p)

CosDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.23 void(* GPUmatNumerics::Cosh)(const GPUtype &p, const GPUtype &q)

Cosh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.24 GPUtype(* GPUmatNumerics::CoshDrv)(const GPUtype &p)

CoshDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.25 void(* GPUmatNumerics::Ctranspose)(const GPUtype &p, const GPUtype &q)

Ctranspose function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.26 GPUtype(* GPUmatNumerics::CtransposeDrv)(const GPUtype &p)

CtransposeDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.27 void(∗ GPUmatNumerics::Eq)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Eq function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.28 GPUtype(∗ GPUmatNumerics::EqDrv)(const GPUtype &p, const GPUtype &q)

EqDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.29 void(∗ GPUmatNumerics::Exp)(const GPUtype &p, const GPUtype &q)

Exp function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.30 GPUtype(∗ GPUmatNumerics::ExpDrv)(const GPUtype &p)

ExpDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.31 void(∗ GPUmatNumerics::Floor)(const GPUtype &p, const GPUtype &q)

Floor function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.32 GPUtype(∗ GPUmatNumerics::FloorDrv)(const GPUtype &p)

FloorDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.33 void(∗ GPUmatNumerics::Ge)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Ge function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.34 GPUtype(∗ GPUmatNumerics::GeDrv)(const GPUtype &p, const GPUtype &q)

GeDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.35 void(∗ GPUmatNumerics::Gt)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Gt function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.36 GPUtype(∗ GPUmatNumerics::GtDrv)(const GPUtype &p, const GPUtype &q)

GtDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.37 void(∗ GPUmatNumerics::Imag)(const GPUtype &p, const GPUtype &q)

Imag function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.38 GPUtype(∗ GPUmatNumerics::ImagDrv)(const GPUtype &p)

ImagDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.39 void(∗ GPUmatNumerics::Ldivide)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Ldivide function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.40 GPUtype(∗ GPUmatNumerics::LdivideDrv)(const GPUtype &p, const GPUtype &q)

LdivideDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.41 void(∗ GPUmatNumerics::Le)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Le function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.42 GPUtype(∗ GPUmatNumerics::LeDrv)(const GPUtype &p, const GPUtype &q)

LeDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.43 void(∗ GPUmatNumerics::Log)(const GPUtype &p, const GPUtype &q)

Log function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.44 void(∗ GPUmatNumerics::Log10)(const GPUtype &p, const GPUtype &q)

Log10 function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.45 GPUtype(∗ GPUmatNumerics::Log10Drv)(const GPUtype &p)

Log10Drv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.46 void(∗ GPUmatNumerics::Log1p)(const GPUtype &p, const GPUtype &q)

Log1p function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.47 GPUtype(* GPUmatNumerics::Log1pDrv)(const GPUtype &p)

Log1pDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.48 void(* GPUmatNumerics::Log2)(const GPUtype &p, const GPUtype &q)

Log2 function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.49 GPUtype(* GPUmatNumerics::Log2Drv)(const GPUtype &p)

Log2Drv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.50 GPUtype(* GPUmatNumerics::LogDrv)(const GPUtype &p)

LogDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.51 void(∗ GPUmatNumerics::Lt)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Lt function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.52 GPUtype(∗ GPUmatNumerics::LtDrv)(const GPUtype &p, const GPUtype &q)

LtDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.53 void(∗ GPUmatNumerics::Minus)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Minus function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.54 GPUtype(∗ GPUmatNumerics::MinusDrv)(const GPUtype &p, const GPUtype &q)

MinusDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.55 void(∗ GPUmatNumerics::Mtimes)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Mtimes function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.56 GPUtype(∗ GPUmatNumerics::MtimesDrv)(const GPUtype &p, const GPUtype &q)

MtimesDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.57 void(∗ GPUmatNumerics::Ne)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Ne function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.58 GPUtype(∗ GPUmatNumerics::NeDrv)(const GPUtype &p, const GPUtype &q)

NeDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.59 void(∗ GPUmatNumerics::Not)(const GPUtype &p, const GPUtype &q)

Not function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.60 GPUtype(∗ GPUmatNumerics::NotDrv)(const GPUtype &p)

NotDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.61 void(∗ GPUmatNumerics::Or)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Or function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.62 GPUtype(∗ GPUmatNumerics::OrDrv)(const GPUtype &p, const GPUtype &q)

OrDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.63 void(* GPUmatNumerics::Plus)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Plus function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.64 GPUtype(* GPUmatNumerics::PlusDrv)(const GPUtype &p, const GPUtype &q)

PlusDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.65 void(* GPUmatNumerics::Power)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Power function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.
→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.66 GPUtype(* GPUmatNumerics::PowerDrv)(const GPUtype &p, const GPUtype &q)

PowerDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.
← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.67 void(∗ GPUmatNumerics::Rdivide)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Rdivide function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.68 GPUtype(∗ GPUmatNumerics::RdivideDrv)(const GPUtype &p, const GPUtype &q)

RdivideDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.69 void(∗ GPUmatNumerics::Real)(const GPUtype &p, const GPUtype &q)

Real function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.70 GPUtype(∗ GPUmatNumerics::RealDrv)(const GPUtype &p)

RealDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.71 void(∗ GPUmatNumerics::Round)(const GPUtype &p, const GPUtype &q)

Round function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.72 GPUtype(∗ GPUmatNumerics::RoundDrv)(const GPUtype &p)

RoundDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.73 void(∗ GPUmatNumerics::Sin)(const GPUtype &p, const GPUtype &q)

Sin function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.74 GPUtype(∗ GPUmatNumerics::SinDrv)(const GPUtype &p)

SinDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.75 void(∗ GPUmatNumerics::Sinh)(const GPUtype &p, const GPUtype &q)

Sinh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.76 GPUtype(∗ GPUmatNumerics::SinhDrv)(const GPUtype &p)

SinhDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.77 void(∗ GPUmatNumerics::Sqrt)(const GPUtype &p, const GPUtype &q)

Sqrt function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.78 GPUtype(∗ GPUmatNumerics::SqrtDrv)(const GPUtype &p)

SqrtDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.79 void(∗ GPUmatNumerics::Tan)(const GPUtype &p, const GPUtype &q)

Tan function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.80 GPUtype(∗ GPUmatNumerics::TanDrv)(const GPUtype &p)

TanDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.81 void(∗ GPUmatNumerics::Tanh)(const GPUtype &p, const GPUtype &q)

Tanh function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.82 GPUtype(∗ GPUmatNumerics::TanhDrv)(const GPUtype &p)

TanhDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.83 void(∗ GPUmatNumerics::Times)(const GPUtype &p, const GPUtype &q, const GPUtype &r)

Times function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

→ ***r*** GPUtype used to store the result.

Returns:

void

2.14.1.84 GPUtype(∗ GPUmatNumerics::TimesDrv)(const GPUtype &p, const GPUtype &q)

TimesDrv function.

Parameters:

← ***p*** 1st GPUtype input variable.

← ***q*** 2nd GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.85 void(∗ GPUmatNumerics::Transpose)(const GPUtype &p, const GPUtype &q)

Transpose function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.86 GPUtype(∗ GPUmatNumerics::TransposeDrv)(const GPUtype &p)

TransposeDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

2.14.1.87 void(* GPUmatNumerics::Uminus)(const GPUtype &p, const GPUtype &q)

Uminus function.

Parameters:

← ***p*** GPUtype input variable.

→ ***q*** GPUtype used to store the result.

Returns:

void

2.14.1.88 GPUtype(* GPUmatNumerics::UminusDrv)(const GPUtype &p)

UminusDrv function.

Parameters:

← ***p*** GPUtype input variable.

Returns:

GPUtype pointer

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

- GPUmatNumerics.hh

2.15 GPUmatRAND Struct Reference

Public Attributes

- void(∗ rand)(const GPUtype &p)
  RAND.
- GPUtype(∗ mxRandDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXRANDDRV.
- void(∗ randn)(const GPUtype &p)
- GPUtype(∗ mxRandnDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])
  MXRANDNDRV.

2.15.1 Member Data Documentation

2.15.1.1 GPUtype(∗ GPUmatRAND::mxRandDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])

MXRANDDRV. Defined in RAND module

2.15.1.2 GPUtype(∗ GPUmatRAND::mxRandnDrv)(const GPUtype &p, int nrhs, const mxArray ∗prhs[ ])

MXRANDNDRV. Defined in RAND module

2.15.1.3 void(∗ GPUmatRAND::rand)(const GPUtype &p)

RAND. Defined in RAND module

2.15.1.4 void(∗ GPUmatRAND::randn)(const GPUtype &p)

Defined in RAND module

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

- GPUmat.hh

2.16 GPUtypeS Struct Reference

Classes

- struct counter

Public Member Functions

- **GPUtypeS** (void *p=NULL, void(*p0)(void *)=NULL)
- **GPUtypeS** (const GPUtypeS &p)
- void acquirePtr (counter *c)

  Increment the count.

- void releasePtr ()

  Decrement the count, delete if 0.

- const GPUtypeS & **operator=** (const GPUtypeS &p)

Public Attributes

- void(* **ptr0**)(void *)
- struct GPUtypeS::counter * **ptrCounter**

2.16.1 Member Function Documentation

2.16.1.1 void GPUtypeS::acquirePtr (counter * *c*) [inline]

Increment the count. acquirePtr

2.16.1.2 void GPUtypeS::releasePtr () [inline]

Decrement the count, delete if 0. releasePtr

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

- GPUmat.hh

2.17 hostdrv_pars Struct Reference

Public Member Functions

- **hostdrv_pars** (void ∗p, int s)
- **hostdrv_pars** (void ∗p, int s, size_t t)

Public Attributes

- void ∗ **par**
- unsigned int **psize**
- size_t **align**

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

- GPUmat.hh

2.18 MyGC Class Reference

Public Member Functions

- void **setPtr** (void ∗p)
- void **remPtr** (void ∗p)

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

- GPUmat.hh

2.19 MyGCObj< C > Class Template Reference

Public Member Functions

- void **setPtr** (C *p)
- void **remPtr** (C *p)

template<class C> class MyGCObj< C >

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

- GPUmat.hh

2.20 RangeS Struct Reference

Public Member Functions

- **RangeS** (int a)
- **RangeS** (int a, const RangeS &r)
- **RangeS** (int a, int b, int c)
- **RangeS** (int a, int b, int c, const RangeS &r)
- **RangeS** (int s, int *c)
- **RangeS** (int s, int *c, const RangeS &r)
- **RangeS** (int s, float *c)
- **RangeS** (int s, float *c, const RangeS &r)
- **RangeS** (int s, double *c)
- **RangeS** (int s, double *c, const RangeS &r)
- **RangeS** (GPUtype &c)
- **RangeS** (GPUtype &c, const RangeS &r)

Public Attributes

- int **inf**
- int **sup**
- int **stride**
- int **begin**
- int **end**
- int * **iindx**
- float * **findx**
- double * **dindx**
- GPUtype * **gindx**
- void * **gindxptr**
- RangeS * **next**

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

- GPUmat.hh

Index