

## Developer Reference

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



# Contents

<b>1</b>	<b>Class Index</b>	<b>1</b>
1.1	Class List . . . . .	1
<b>2</b>	<b>Class Documentation</b>	<b>3</b>
2.1	GPUtypeS::counter Struct Reference . . . . .	3
2.2	GPUexception Class Reference . . . . .	4
2.3	GPUmat Struct Reference . . . . .	5
2.4	GPUmatAux Struct Reference . . . . .	6
2.4.1	Member Data Documentation . . . . .	6
2.4.1.1	mxAssign . . . . .	6
2.4.1.2	mxSliceDrv . . . . .	6
2.5	GPUmatCompiler Struct Reference . . . . .	7
2.5.1	Member Data Documentation . . . . .	8
2.5.1.1	abort . . . . .	8
2.5.1.2	createMxContext . . . . .	8
2.5.1.3	getCompileMode . . . . .	8
2.5.1.4	getContextGPUtype . . . . .	8
2.5.1.5	getContextMx . . . . .	8
2.5.1.6	pushGPUtype . . . . .	8
2.5.1.7	pushMx . . . . .	8
2.5.1.8	registerInstruction . . . . .	9
2.6	GPUmatDebug Struct Reference . . . . .	10
2.6.1	Member Data Documentation . . . . .	10
2.6.1.1	debugPushInstructionStack . . . . .	10
2.6.1.2	getDebugMode . . . . .	10
2.6.1.3	log . . . . .	10
2.6.1.4	logPop . . . . .	10
2.6.1.5	logPush . . . . .	11

2.6.1.6	setDebugMode	11
2.7	GPUMatFFT Struct Reference	12
2.7.1	Member Data Documentation	12
2.7.1.1	FFT1Drv	12
2.7.1.2	FFT2Drv	12
2.7.1.3	FFT3Drv	13
2.7.1.4	IFFT1Drv	13
2.7.1.5	IFFT2Drv	13
2.7.1.6	IFFT3Drv	13
2.8	GPUMatGPUtype Struct Reference	14
2.8.1	Member Data Documentation	16
2.8.1.1	assign	16
2.8.1.2	clone	17
2.8.1.3	colon	17
2.8.1.4	create	17
2.8.1.5	createMx	18
2.8.1.6	createMxArray	18
2.8.1.7	doubleToFloat	18
2.8.1.8	eye	18
2.8.1.9	fill	19
2.8.1.10	floatToDouble	19
2.8.1.11	getDataSize	19
2.8.1.12	getGPUptr	19
2.8.1.13	getGPUtype	20
2.8.1.14	getNdims	20
2.8.1.15	getNumel	20
2.8.1.16	getSize	20
2.8.1.17	getType	21
2.8.1.18	isComplex	21
2.8.1.19	isDouble	21
2.8.1.20	isEmpty	21
2.8.1.21	isFloat	22
2.8.1.22	isScalar	22
2.8.1.23	mxAssign	22
2.8.1.24	mxColonDrv	22
2.8.1.25	mxEyeDrv	22

2.8.1.26	<a href="#">mxFill</a>	23
2.8.1.27	<a href="#">mxMemCpyDtoD</a>	23
2.8.1.28	<a href="#">mxMemCpyHtoD</a>	23
2.8.1.29	<a href="#">mxOnesDrv</a>	23
2.8.1.30	<a href="#">mxPermute</a>	23
2.8.1.31	<a href="#">mxPermuteDrv</a>	23
2.8.1.32	<a href="#">mxRepmatDrv</a>	23
2.8.1.33	<a href="#">mxSlice</a>	24
2.8.1.34	<a href="#">mxToGPUtype</a>	24
2.8.1.35	<a href="#">mxZerosDrv</a>	24
2.8.1.36	<a href="#">ones</a>	24
2.8.1.37	<a href="#">permute</a>	24
2.8.1.38	<a href="#">realimag</a>	25
2.8.1.39	<a href="#">realImagToComplex</a>	25
2.8.1.40	<a href="#">realToComplex</a>	25
2.8.1.41	<a href="#">setSize</a>	25
2.8.1.42	<a href="#">slice</a>	26
2.8.1.43	<a href="#">toMxArray</a>	26
2.8.1.44	<a href="#">zeros</a>	26
2.9	<a href="#">GPUmatInterface Struct Reference</a>	27
2.10	<a href="#">GPUmatInterfaceConfig Struct Reference</a>	28
2.10.1	<a href="#">Member Data Documentation</a>	28
2.10.1.1	<a href="#">getActiveDeviceNumber</a>	28
2.10.1.2	<a href="#">getMajorMinor</a>	28
2.11	<a href="#">GPUmatInterfaceFunction Struct Reference</a>	29
2.11.1	<a href="#">Member Data Documentation</a>	29
2.11.1.1	<a href="#">getFunctionByName</a>	29
2.11.1.2	<a href="#">getFunctionByNumber</a>	29
2.11.1.3	<a href="#">getFunctionNumber</a>	29
2.11.1.4	<a href="#">registerFunction</a>	30
2.12	<a href="#">GPUmatManager Struct Reference</a>	31
2.12.1	<a href="#">Member Data Documentation</a>	31
2.12.1.1	<a href="#">cacheClean</a>	31
2.13	<a href="#">GPUmatModules Struct Reference</a>	32
2.14	<a href="#">GPUmatNumerics Struct Reference</a>	33
2.14.1	<a href="#">Member Data Documentation</a>	38

2.14.1.1	Abs	38
2.14.1.2	AbsDrv	38
2.14.1.3	Acos	39
2.14.1.4	AcosDrv	39
2.14.1.5	Acosh	39
2.14.1.6	AcoshDrv	39
2.14.1.7	And	40
2.14.1.8	AndDrv	40
2.14.1.9	Asin	40
2.14.1.10	AsinDrv	40
2.14.1.11	Asinh	41
2.14.1.12	AsinhDrv	41
2.14.1.13	Atan	41
2.14.1.14	AtanDrv	41
2.14.1.15	Atanh	42
2.14.1.16	AtanhDrv	42
2.14.1.17	Ceil	42
2.14.1.18	CeilDrv	42
2.14.1.19	Conj	43
2.14.1.20	ConjDrv	43
2.14.1.21	Cos	43
2.14.1.22	CosDrv	43
2.14.1.23	Cosh	44
2.14.1.24	CoshDrv	44
2.14.1.25	Ctranspose	44
2.14.1.26	CtransposeDrv	44
2.14.1.27	Eq	45
2.14.1.28	EqDrv	45
2.14.1.29	Exp	45
2.14.1.30	ExpDrv	45
2.14.1.31	Floor	46
2.14.1.32	FloorDrv	46
2.14.1.33	Ge	46
2.14.1.34	GeDrv	46
2.14.1.35	Gt	47
2.14.1.36	GtDrv	47

2.14.1.37 Imag	47
2.14.1.38 ImagDrv	47
2.14.1.39 Ldivide	48
2.14.1.40 LdivideDrv	48
2.14.1.41 Le	48
2.14.1.42 LeDrv	48
2.14.1.43 Log	49
2.14.1.44 Log10	49
2.14.1.45 Log10Drv	49
2.14.1.46 Log1p	49
2.14.1.47 Log1pDrv	50
2.14.1.48 Log2	50
2.14.1.49 Log2Drv	50
2.14.1.50 LogDrv	50
2.14.1.51 Lt	51
2.14.1.52 LtDrv	51
2.14.1.53 Minus	51
2.14.1.54 MinusDrv	51
2.14.1.55 Mtimes	52
2.14.1.56 MtimesDrv	52
2.14.1.57 Ne	52
2.14.1.58 NeDrv	52
2.14.1.59 Not	53
2.14.1.60 NotDrv	53
2.14.1.61 Or	53
2.14.1.62 OrDrv	53
2.14.1.63 Plus	54
2.14.1.64 PlusDrv	54
2.14.1.65 Power	54
2.14.1.66 PowerDrv	54
2.14.1.67 Rdivide	55
2.14.1.68 RdivideDrv	55
2.14.1.69 Real	55
2.14.1.70 RealDrv	55
2.14.1.71 Round	56
2.14.1.72 RoundDrv	56

2.14.1.73 Sin	56
2.14.1.74 SinDrv	56
2.14.1.75 Sinh	57
2.14.1.76 SinhDrv	57
2.14.1.77 Sqrt	57
2.14.1.78 SqrtDrv	57
2.14.1.79 Tan	58
2.14.1.80 TanDrv	58
2.14.1.81 Tanh	58
2.14.1.82 TanhDrv	58
2.14.1.83 Times	59
2.14.1.84 TimesDrv	59
2.14.1.85 Transpose	59
2.14.1.86 TransposeDrv	59
2.14.1.87 Uminus	60
2.14.1.88 UminusDrv	60
2.15 GPUmatRAND Struct Reference	61
2.15.1 Member Data Documentation	61
2.15.1.1 mxRandDrv	61
2.15.1.2 mxRandnDrv	61
2.15.1.3 rand	61
2.15.1.4 randn	61
2.16 GPUtypeS Struct Reference	62
2.16.1 Member Function Documentation	62
2.16.1.1 acquirePtr	62
2.16.1.2 releasePtr	62
2.17 hostdrv_pars Struct Reference	63
2.18 MyGC Class Reference	64
2.19 MyGCObj< C > Class Template Reference	65
2.20 RangeS Struct Reference	66



# Chapter 1

## Class Index

### 1.1 Class List

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

<a href="#">GPUtypeS::counter</a>	3
<a href="#">GPUexception</a>	4
<a href="#">GPUmat</a>	5
<a href="#">GPUmatAux</a>	6
<a href="#">GPUmatCompiler</a>	7
<a href="#">GPUmatDebug</a>	10
<a href="#">GPUmatFFT</a>	12
<a href="#">GPUmatGPUtype</a>	14
<a href="#">GPUmatInterface</a>	27
<a href="#">GPUmatInterfaceConfig</a>	28
<a href="#">GPUmatInterfaceFunction</a>	29
<a href="#">GPUmatManager</a>	31
<a href="#">GPUmatModules</a>	32
<a href="#">GPUmatNumerics</a>	33
<a href="#">GPUmatRAND</a>	61
<a href="#">GPUtypeS</a>	62
<a href="#">hostdrv_pars</a>	63
<a href="#">MyGC</a>	64
<a href="#">MyGCObj&lt; C &gt;</a>	65
<a href="#">RangeS</a>	66



## 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 in 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 in 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:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

A new GPUtype object

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

IFFT1D.

**Parameters:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

A new GPUtype object

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

IFFT2D.

**Parameters:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

A new GPUtype object

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

IFFT3D.

**Parameters:**

$\leftarrow p$  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  $\text{incr} * (i \% m) + \text{offset}$  if  $((i + \text{offsetp}) \% p) == 0$ ;

#### Parameters:

- $\leftarrow q$  GPUtype.
- $\leftarrow \text{offset}$  (see formula above)
- $\leftarrow \text{incr}$  (see formula above)
- $\leftarrow m$  (see formula above)
- $\leftarrow p$  (see formula above)
- $\leftarrow \text{offsetp}$  (see formula above)
- $\leftarrow \text{type}=0$  only real part is modified  $\text{type}=1$  only imaginary part is modified  $\text{type}=2$  both real and imaginary are modified

### 2.8.1.10 GPUtype(\* GPUmatGPUtype::floatToDouble)(const GPUtype &p)

Cast from FLOAT to DOUBLE.

#### Parameters:

- $\leftarrow p$  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:

- $\leftarrow p$  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:

- $\leftarrow p$  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:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

gpuTYPE\_t

**2.8.1.18** `int(* GPUmatGPUtype::isComplex)(const GPUtype &p)`

Returns 1 if the GPUtype is COMPLEX.

**Parameters:**

$\leftarrow p$  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:**

$\leftarrow p$  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:**

$\leftarrow p$  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:

- $\leftarrow p$  GPUtype input variable
- $\leftarrow r$  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:

- $\leftarrow p$  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:

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

#### Returns:

void

### 2.14.1.2 `GPUtype(* GPUmatNumerics::AbsDrv)(const GPUtype &p)`

AbsDrv function.

#### Parameters:

- $\leftarrow p$  GPUtype input variable.

#### Returns:

GPUtype pointer

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

Acos function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.4 GPUtype(\* GPUmatNumerics::AcosDrv)(const GPUtype &p)**

AcosDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Acosh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.6 GPUtype(\* GPUmatNumerics::AcoshDrv)(const GPUtype &p)**

AcoshDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

#### 2.14.1.8 GPUtype(\* GPUmatNumerics::AndDrv)(const GPUtype &p, const GPUtype &q)

AndDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

#### 2.14.1.9 void(\* GPUmatNumerics::Asin)(const GPUtype &p, const GPUtype &q)

Asin function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

#### 2.14.1.10 GPUtype(\* GPUmatNumerics::AsinDrv)(const GPUtype &p)

AsinDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer



**2.14.1.11 void(\* GPUMatNumerics::Asinh)(const GPUtype &p, const GPUtype &q)**

Asinh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.12 GPUtype(\* GPUMatNumerics::AsinhDrv)(const GPUtype &p)**

AsinhDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

**2.14.1.13 void(\* GPUMatNumerics::Atan)(const GPUtype &p, const GPUtype &q)**

Atan function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.14 GPUtype(\* GPUMatNumerics::AtanDrv)(const GPUtype &p)**

AtanDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Atanh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

AtanhDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Ceil function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

CeilDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Conj function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.20 GPUtype(\* GPUmatNumerics::ConjDrv)(const GPUtype &p)**

ConjDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Cos function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.22 GPUtype(\* GPUmatNumerics::CosDrv)(const GPUtype &p)**

CosDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Cosh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

CoshDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Ctranspose function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

CtransposeDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

EqDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

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

Exp function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.30 GPUtype(\* GPUmatNumerics::ExpDrv)(const GPUtype &p)**

ExpDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Floor function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.32 GPUtype(\* GPUmatNumerics::FloorDrv)(const GPUtype &p)**

FloorDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

GeDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

GtDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

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

Imag function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.38 GPUtype(\* GPUmatNumerics::ImagDrv)(const GPUtype &p)**

ImagDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

Log10 function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.45 GPUtype(\* GPUmatNumerics::Log10Drv)(const GPUtype &p)**

Log10Drv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Log1p function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

Log1pDrv function.

**Parameters:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Log2 function.

**Parameters:**

$\leftarrow p$  GPUtype input variable.

$\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

Log2Drv function.

**Parameters:**

$\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

LogDrv function.

**Parameters:**

$\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

LtDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

MinusDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

MtimesDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

NeDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

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

Not function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.60 GPUtype(\* GPUmatNumerics::NotDrv)(const GPUtype &p)**

NotDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

OrDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

PlusDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

PowerDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.68 GPUtype(\* GPUMatNumerics::RdivideDrv)(const GPUtype &p, const GPUtype &q)**

RdivideDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

**2.14.1.69 void(\* GPUMatNumerics::Real)(const GPUtype &p, const GPUtype &q)**

Real function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.70 GPUtype(\* GPUMatNumerics::RealDrv)(const GPUtype &p)**

RealDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Round function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.72 GPUtype(\* GPUmatNumerics::RoundDrv)(const GPUtype &p)**

RoundDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Sin function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.74 GPUtype(\* GPUmatNumerics::SinDrv)(const GPUtype &p)**

SinDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer



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

Sinh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.76 GPUtype(\* GPUmatNumerics::SinhDrv)(const GPUtype &p)**

SinhDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Sqrt function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.78 GPUtype(\* GPUmatNumerics::SqrtDrv)(const GPUtype &p)**

SqrtDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Tan function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.80 GPUtype(\* GPUmatNumerics::TanDrv)(const GPUtype &p)**

TanDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Tanh function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.82 GPUtype(\* GPUmatNumerics::TanhDrv)(const GPUtype &p)**

TanhDrv function.

**Parameters:**

- $\leftarrow p$  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:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.
- $\rightarrow r$  GPUtype used to store the result.

**Returns:**

void

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

TimesDrv function.

**Parameters:**

- $\leftarrow p$  1st GPUtype input variable.
- $\leftarrow q$  2nd GPUtype input variable.

**Returns:**

GPUtype pointer

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

Transpose function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

**2.14.1.86 GPUtype(\* GPUmatNumerics::TransposeDrv)(const GPUtype &p)**

TransposeDrv function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.

**Returns:**

GPUtype pointer

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

Uminus function.

**Parameters:**

- $\leftarrow p$  GPUtype input variable.
- $\rightarrow q$  GPUtype used to store the result.

**Returns:**

void

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

UminusDrv function.

**Parameters:**

- $\leftarrow p$  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

- abort
  - GPUMatCompiler, [8](#)
- Abs
  - GPUMatNumerics, [38](#)
- AbsDrv
  - GPUMatNumerics, [38](#)
- Acos
  - GPUMatNumerics, [38](#)
- AcosDrv
  - GPUMatNumerics, [39](#)
- Acosh
  - GPUMatNumerics, [39](#)
- AcoshDrv
  - GPUMatNumerics, [39](#)
- acquirePtr
  - GPUtypeS, [62](#)
- And
  - GPUMatNumerics, [39](#)
- AndDrv
  - GPUMatNumerics, [40](#)
- Asin
  - GPUMatNumerics, [40](#)
- AsinDrv
  - GPUMatNumerics, [40](#)
- Asinh
  - GPUMatNumerics, [40](#)
- AsinhDrv
  - GPUMatNumerics, [41](#)
- assign
  - GPUMatGPUtype, [16](#)
- Atan
  - GPUMatNumerics, [41](#)
- AtanDrv
  - GPUMatNumerics, [41](#)
- Atanh
  - GPUMatNumerics, [41](#)
- AtanhDrv
  - GPUMatNumerics, [42](#)
- cacheClean
  - GPUMatManager, [31](#)
- Ceil
  - GPUMatNumerics, [42](#)
- CeilDrv
  - GPUMatNumerics, [42](#)
- clone
  - GPUMatGPUtype, [17](#)
- colon
  - GPUMatGPUtype, [17](#)
- Conj
  - GPUMatNumerics, [42](#)
- ConjDrv
  - GPUMatNumerics, [43](#)
- Cos
  - GPUMatNumerics, [43](#)
- CosDrv
  - GPUMatNumerics, [43](#)
- Cosh
  - GPUMatNumerics, [43](#)
- CoshDrv
  - GPUMatNumerics, [44](#)
- create
  - GPUMatGPUtype, [17](#)
- createMx
  - GPUMatGPUtype, [17](#)
- createMxArray
  - GPUMatGPUtype, [18](#)
- createMxContext
  - GPUMatCompiler, [8](#)
- Ctranspose
  - GPUMatNumerics, [44](#)
- CtransposeDrv
  - GPUMatNumerics, [44](#)
- debugPushInstructionStack
  - GPUMatDebug, [10](#)
- doubleToFloat
  - GPUMatGPUtype, [18](#)
- Eq
  - GPUMatNumerics, [44](#)
- EqDrv
  - GPUMatNumerics, [45](#)
- Exp
  - GPUMatNumerics, [45](#)
- ExpDrv
  - GPUMatNumerics, [45](#)
- eye
  - GPUMatGPUtype, [18](#)
- FFT1Drv

- GPUmatFFT, 12
- FFT2Drv
  - GPUmatFFT, 12
- FFT3Drv
  - GPUmatFFT, 12
- fill
  - GPUmatGPUtype, 18
- floatToDouble
  - GPUmatGPUtype, 19
- Floor
  - GPUmatNumerics, 45
- FloorDrv
  - GPUmatNumerics, 46
- Ge
  - GPUmatNumerics, 46
- GeDrv
  - GPUmatNumerics, 46
- getActiveDeviceNumber
  - GPUmatInterfaceConfig, 28
- getCompileMode
  - GPUmatCompiler, 8
- getContextGPUtype
  - GPUmatCompiler, 8
- getContextMx
  - GPUmatCompiler, 8
- getDataSize
  - GPUmatGPUtype, 19
- getDebugMode
  - GPUmatDebug, 10
- getFunctionByName
  - GPUmatInterfaceFunction, 29
- getFunctionByNumber
  - GPUmatInterfaceFunction, 29
- getFunctionNumber
  - GPUmatInterfaceFunction, 29
- getGPUptr
  - GPUmatGPUtype, 19
- getGPUtype
  - GPUmatGPUtype, 19
- getMajorMinor
  - GPUmatInterfaceConfig, 28
- getNdims
  - GPUmatGPUtype, 20
- getNumel
  - GPUmatGPUtype, 20
- getSize
  - GPUmatGPUtype, 20
- getType
  - GPUmatGPUtype, 20
- GPUexception, 4
- GPUmat, 5
- GPUmatAux, 6
  - mxAssign, 6
- mxSliceDrv, 6
- GPUmatCompiler, 7
  - abort, 8
  - createMxContext, 8
  - getCompileMode, 8
  - getContextGPUtype, 8
  - getContextMx, 8
  - pushGPUtype, 8
  - pushMx, 8
  - registerInstruction, 8
- GPUmatDebug, 10
  - debugPushInstructionStack, 10
  - getDebugMode, 10
  - log, 10
  - logPop, 10
  - logPush, 10
  - setDebugMode, 11
- GPUmatFFT, 12
  - FFT1Drv, 12
  - FFT2Drv, 12
  - FFT3Drv, 12
  - IFFT1Drv, 13
  - IFFT2Drv, 13
  - IFFT3Drv, 13
- GPUmatGPUtype, 14
  - assign, 16
  - clone, 17
  - colon, 17
  - create, 17
  - createMx, 17
  - createMxArray, 18
  - doubleToFloat, 18
  - eye, 18
  - fill, 18
  - floatToDouble, 19
  - getDataSize, 19
  - getGPUptr, 19
  - getGPUtype, 19
  - getNdims, 20
  - getNumel, 20
  - getSize, 20
  - getType, 20
  - isComplex, 21
  - isDouble, 21
  - isEmpty, 21
  - isFloat, 21
  - isScalar, 22
  - mxAssign, 22
  - mxColonDrv, 22
  - mxEyeDrv, 22
  - mxFill, 22
  - mxMemCpyDtoD, 23
  - mxMemCpyHtoD, 23
  - mxOnesDrv, 23

- mxPermute, 23
- mxPermuteDrv, 23
- mxRepmatDrv, 23
- mxSlice, 23
- mxToGPUtype, 24
- mxZerosDrv, 24
- ones, 24
- permute, 24
- realimag, 24
- realImagToComplex, 25
- realToComplex, 25
- setSize, 25
- slice, 25
- toMxArray, 26
- zeros, 26
- GPUmatInterface, 27
- GPUmatInterfaceConfig, 28
  - getActiveDeviceNumber, 28
  - getMajorMinor, 28
- GPUmatInterfaceFunction, 29
  - getFunctionByName, 29
  - getFunctionByNumber, 29
  - getFunctionNumber, 29
  - registerFunction, 29
- GPUmatManager, 31
  - cacheClean, 31
- GPUmatModules, 32
- GPUmatNumerics, 33
  - Abs, 38
  - AbsDrv, 38
  - Acos, 38
  - AcosDrv, 39
  - Acosh, 39
  - AcoshDrv, 39
  - And, 39
  - AndDrv, 40
  - Asin, 40
  - AsinDrv, 40
  - Asinh, 40
  - AsinhDrv, 41
  - Atan, 41
  - AtanDrv, 41
  - Atanh, 41
  - AtanhDrv, 42
  - Ceil, 42
  - CeilDrv, 42
  - Conj, 42
  - ConjDrv, 43
  - Cos, 43
  - CosDrv, 43
  - Cosh, 43
  - CoshDrv, 44
  - Ctranspose, 44
  - CtransposeDrv, 44
  - Eq, 44
  - EqDrv, 45
  - Exp, 45
  - ExpDrv, 45
  - Floor, 45
  - FloorDrv, 46
  - Ge, 46
  - GeDrv, 46
  - Gt, 46
  - GtDrv, 47
  - Imag, 47
  - ImagDrv, 47
  - Ldivide, 47
  - LdivideDrv, 48
  - Le, 48
  - LeDrv, 48
  - Log, 48
  - Log10, 49
  - Log10Drv, 49
  - Log1p, 49
  - Log1pDrv, 49
  - Log2, 50
  - Log2Drv, 50
  - LogDrv, 50
  - Lt, 50
  - LtDrv, 51
  - Minus, 51
  - MinusDrv, 51
  - Mtimes, 51
  - MtimesDrv, 52
  - Ne, 52
  - NeDrv, 52
  - Not, 52
  - NotDrv, 53
  - Or, 53
  - OrDrv, 53
  - Plus, 53
  - PlusDrv, 54
  - Power, 54
  - PowerDrv, 54
  - Rdivide, 54
  - RdivideDrv, 55
  - Real, 55
  - RealDrv, 55
  - Round, 55
  - RoundDrv, 56
  - Sin, 56
  - SinDrv, 56
  - Sinh, 56
  - SinhDrv, 57
  - Sqrt, 57
  - SqrtDrv, 57
  - Tan, 57
  - TanDrv, 58

- Tanh, [58](#)
- TanhDrv, [58](#)
- Times, [58](#)
- TimesDrv, [59](#)
- Transpose, [59](#)
- TransposeDrv, [59](#)
- Uminus, [59](#)
- UminusDrv, [60](#)
- GPUmatRAND, [61](#)
  - mxRandDrv, [61](#)
  - mxRandnDrv, [61](#)
  - rand, [61](#)
  - randn, [61](#)
- GPUtypeS, [62](#)
  - acquirePtr, [62](#)
  - releasePtr, [62](#)
- GPUtypeS::counter, [3](#)
- Gt
  - GPUmatNumerics, [46](#)
- GtDrv
  - GPUmatNumerics, [47](#)
- hostdrv\_pars, [63](#)
- IFFT1Drv
  - GPUmatFFT, [13](#)
- IFFT2Drv
  - GPUmatFFT, [13](#)
- IFFT3Drv
  - GPUmatFFT, [13](#)
- Imag
  - GPUmatNumerics, [47](#)
- ImagDrv
  - GPUmatNumerics, [47](#)
- isComplex
  - GPUmatGPUtype, [21](#)
- isDouble
  - GPUmatGPUtype, [21](#)
- isEmpty
  - GPUmatGPUtype, [21](#)
- isFloat
  - GPUmatGPUtype, [21](#)
- isScalar
  - GPUmatGPUtype, [22](#)
- Ldivide
  - GPUmatNumerics, [47](#)
- LdivideDrv
  - GPUmatNumerics, [48](#)
- Le
  - GPUmatNumerics, [48](#)
- LeDrv
  - GPUmatNumerics, [48](#)
- Log
  - GPUmatNumerics, [48](#)
- log
  - GPUmatDebug, [10](#)
- Log10
  - GPUmatNumerics, [49](#)
- Log10Drv
  - GPUmatNumerics, [49](#)
- Log1p
  - GPUmatNumerics, [49](#)
- Log1pDrv
  - GPUmatNumerics, [49](#)
- Log2
  - GPUmatNumerics, [50](#)
- Log2Drv
  - GPUmatNumerics, [50](#)
- LogDrv
  - GPUmatNumerics, [50](#)
- logPop
  - GPUmatDebug, [10](#)
- logPush
  - GPUmatDebug, [10](#)
- Lt
  - GPUmatNumerics, [50](#)
- LtDrv
  - GPUmatNumerics, [51](#)
- Minus
  - GPUmatNumerics, [51](#)
- MinusDrv
  - GPUmatNumerics, [51](#)
- Mtimes
  - GPUmatNumerics, [51](#)
- MtimesDrv
  - GPUmatNumerics, [52](#)
- mxAssign
  - GPUmatAux, [6](#)
  - GPUmatGPUtype, [22](#)
- mxColonDrv
  - GPUmatGPUtype, [22](#)
- mxEyeDrv
  - GPUmatGPUtype, [22](#)
- mxFill
  - GPUmatGPUtype, [22](#)
- mxMemCpyDtoD
  - GPUmatGPUtype, [23](#)
- mxMemCpyHtoD
  - GPUmatGPUtype, [23](#)
- mxOnesDrv
  - GPUmatGPUtype, [23](#)
- mxPermute
  - GPUmatGPUtype, [23](#)
- mxPermuteDrv
  - GPUmatGPUtype, [23](#)
- mxRandDrv

- GPUmatRAND, 61
- mxRandnDrv
  - GPUmatRAND, 61
- mxRepmatDrv
  - GPUmatGPUtype, 23
- mxSlice
  - GPUmatGPUtype, 23
- mxSliceDrv
  - GPUmatAux, 6
- mxToGPUtype
  - GPUmatGPUtype, 24
- mxZerosDrv
  - GPUmatGPUtype, 24
- MyGC, 64
- MyGCObj, 65
- Ne
  - GPUmatNumerics, 52
- NeDrv
  - GPUmatNumerics, 52
- Not
  - GPUmatNumerics, 52
- NotDrv
  - GPUmatNumerics, 53
- ones
  - GPUmatGPUtype, 24
- Or
  - GPUmatNumerics, 53
- OrDrv
  - GPUmatNumerics, 53
- permute
  - GPUmatGPUtype, 24
- Plus
  - GPUmatNumerics, 53
- PlusDrv
  - GPUmatNumerics, 54
- Power
  - GPUmatNumerics, 54
- PowerDrv
  - GPUmatNumerics, 54
- pushGPUtype
  - GPUmatCompiler, 8
- pushMx
  - GPUmatCompiler, 8
- rand
  - GPUmatRAND, 61
- randn
  - GPUmatRAND, 61
- RangeS, 66
- Rdivide
  - GPUmatNumerics, 54
- RdivideDrv
  - GPUmatNumerics, 55
- Real
  - GPUmatNumerics, 55
- RealDrv
  - GPUmatNumerics, 55
- realimag
  - GPUmatGPUtype, 24
- realImagToComplex
  - GPUmatGPUtype, 25
- realToComplex
  - GPUmatGPUtype, 25
- registerFunction
  - GPUmatInterfaceFunction, 29
- registerInstruction
  - GPUmatCompiler, 8
- releasePtr
  - GPUtypeS, 62
- Round
  - GPUmatNumerics, 55
- RoundDrv
  - GPUmatNumerics, 56
- setDebugMode
  - GPUmatDebug, 11
- setSize
  - GPUmatGPUtype, 25
- Sin
  - GPUmatNumerics, 56
- SinDrv
  - GPUmatNumerics, 56
- Sinh
  - GPUmatNumerics, 56
- SinhDrv
  - GPUmatNumerics, 57
- slice
  - GPUmatGPUtype, 25
- Sqrt
  - GPUmatNumerics, 57
- SqrtDrv
  - GPUmatNumerics, 57
- Tan
  - GPUmatNumerics, 57
- TanDrv
  - GPUmatNumerics, 58
- Tanh
  - GPUmatNumerics, 58
- TanhDrv
  - GPUmatNumerics, 58
- Times
  - GPUmatNumerics, 58
- TimesDrv
  - GPUmatNumerics, 59

toMxArray  
    GPUmatGPUtype, [26](#)  
Transpose  
    GPUmatNumerics, [59](#)  
TransposeDrv  
    GPUmatNumerics, [59](#)  
  
Uminus  
    GPUmatNumerics, [59](#)  
UminusDrv  
    GPUmatNumerics, [60](#)  
  
zeros  
    GPUmatGPUtype, [26](#)