

# GPUmat User Modules Reference

Generated by Doxygen 1.6.1

Mon Apr 19 16:04:21 2010



# Contents

0.1	GPUmat Struct Reference	1
0.2	GPUmatCompiler Struct Reference	2
0.2.1	Member Data Documentation	3
0.2.1.1	abort	3
0.2.1.2	createMxContext	3
0.2.1.3	getCompileMode	3
0.2.1.4	getContextGPUtype	3
0.2.1.5	getContextMx	3
0.2.1.6	pushGPUtype	3
0.2.1.7	pushMx	3
0.2.1.8	registerInstruction	4
0.3	GPUmatDebug Struct Reference	5
0.3.1	Member Data Documentation	5
0.3.1.1	debugPushInstructionStack	5
0.3.1.2	getDebugMode	5
0.3.1.3	log	5
0.3.1.4	logPop	5
0.3.1.5	logPush	6
0.3.1.6	setDebugMode	6
0.4	GPUmatFFT Struct Reference	7
0.4.1	Member Data Documentation	7
0.4.1.1	FFT1Drv	7
0.4.1.2	FFT2Drv	7
0.4.1.3	FFT3Drv	8
0.4.1.4	IFFT1Drv	8
0.4.1.5	IFFT2Drv	8
0.4.1.6	IFFT3Drv	8
0.5	GPUmatGPUtype Struct Reference	9

---

0.5.1	Member Data Documentation	11
0.5.1.1	assign	11
0.5.1.2	clone	12
0.5.1.3	colon	12
0.5.1.4	create	12
0.5.1.5	createMx	13
0.5.1.6	createMxArray	13
0.5.1.7	doubleToFloat	13
0.5.1.8	eye	13
0.5.1.9	fill	14
0.5.1.10	floatToDouble	14
0.5.1.11	getDataSize	14
0.5.1.12	getGPUptr	14
0.5.1.13	getGPUtype	15
0.5.1.14	getNdims	15
0.5.1.15	getNumel	15
0.5.1.16	getSize	15
0.5.1.17	getType	16
0.5.1.18	isComplex	16
0.5.1.19	isDouble	16
0.5.1.20	isEmpty	16
0.5.1.21	isFloat	17
0.5.1.22	isScalar	17
0.5.1.23	mxAssign	17
0.5.1.24	mxColonDrv	17
0.5.1.25	mxEyeDrv	17
0.5.1.26	mxFill	18
0.5.1.27	mxMemCpyDtoD	18
0.5.1.28	mxMemCpyHtoD	18
0.5.1.29	mxOnesDrv	18
0.5.1.30	mxPermute	18
0.5.1.31	mxPermuteDrv	18
0.5.1.32	mxRepmatDrv	18
0.5.1.33	mxSlice	19
0.5.1.34	mxToGPUtype	19
0.5.1.35	mxZerosDrv	19

---

0.5.1.36	ones	19
0.5.1.37	permute	19
0.5.1.38	realimag	20
0.5.1.39	realImagToComplex	20
0.5.1.40	realToComplex	20
0.5.1.41	setSize	20
0.5.1.42	slice	21
0.5.1.43	toMxArray	21
0.5.1.44	zeros	21
0.6	GPUmatInterface Struct Reference	22
0.7	GPUmatInterfaceConfig Struct Reference	23
0.7.1	Member Data Documentation	23
0.7.1.1	getMajorMinor	23
0.8	GPUmatInterfaceFunction Struct Reference	24
0.8.1	Member Data Documentation	24
0.8.1.1	getFunctionByName	24
0.8.1.2	getFunctionByNumber	24
0.8.1.3	getFunctionNumber	24
0.8.1.4	registerFunction	25
0.9	GPUmatModules Struct Reference	26
0.10	GPUmatNumerics Struct Reference	27
0.10.1	Member Data Documentation	32
0.10.1.1	Abs	32
0.10.1.2	AbsDrv	32
0.10.1.3	Acos	33
0.10.1.4	AcosDrv	33
0.10.1.5	Acosh	33
0.10.1.6	AcoshDrv	33
0.10.1.7	And	34
0.10.1.8	AndDrv	34
0.10.1.9	Asin	34
0.10.1.10	AsinDrv	34
0.10.1.11	Asinh	35
0.10.1.12	AsinhDrv	35
0.10.1.13	Atan	35
0.10.1.14	AtanDrv	35

---

0.10.1.15 Atanh	36
0.10.1.16 AtanhDrv	36
0.10.1.17 Ceil	36
0.10.1.18 CeilDrv	36
0.10.1.19 Conj	37
0.10.1.20 ConjDrv	37
0.10.1.21 Cos	37
0.10.1.22 CosDrv	37
0.10.1.23 Cosh	38
0.10.1.24 CoshDrv	38
0.10.1.25 Ctranspose	38
0.10.1.26 CtransposeDrv	38
0.10.1.27 Eq	39
0.10.1.28 EqDrv	39
0.10.1.29 Exp	39
0.10.1.30 ExpDrv	39
0.10.1.31 Floor	40
0.10.1.32 FloorDrv	40
0.10.1.33 Ge	40
0.10.1.34 GeDrv	40
0.10.1.35 Gt	41
0.10.1.36 GtDrv	41
0.10.1.37 Imag	41
0.10.1.38 ImagDrv	41
0.10.1.39 Ldivide	42
0.10.1.40 LdivideDrv	42
0.10.1.41 Le	42
0.10.1.42 LeDrv	42
0.10.1.43 Log	43
0.10.1.44 Log10	43
0.10.1.45 Log10Drv	43
0.10.1.46 Log1p	43
0.10.1.47 Log1pDrv	44
0.10.1.48 Log2	44
0.10.1.49 Log2Drv	44
0.10.1.50 LogDrv	44

---

0.10.1.51 Lt	45
0.10.1.52 LtDrv	45
0.10.1.53 Minus	45
0.10.1.54 MinusDrv	45
0.10.1.55 Mtimes	46
0.10.1.56 MtimesDrv	46
0.10.1.57 Ne	46
0.10.1.58 NeDrv	46
0.10.1.59 Not	47
0.10.1.60 NotDrv	47
0.10.1.61 Or	47
0.10.1.62 OrDrv	47
0.10.1.63 Plus	48
0.10.1.64 PlusDrv	48
0.10.1.65 Power	48
0.10.1.66 PowerDrv	48
0.10.1.67 Rdivide	49
0.10.1.68 RdivideDrv	49
0.10.1.69 Real	49
0.10.1.70 RealDrv	49
0.10.1.71 Round	50
0.10.1.72 RoundDrv	50
0.10.1.73 Sin	50
0.10.1.74 SinDrv	50
0.10.1.75 Sinh	51
0.10.1.76 SinhDrv	51
0.10.1.77 Sqrt	51
0.10.1.78 SqrtDrv	51
0.10.1.79 Tan	52
0.10.1.80 TanDrv	52
0.10.1.81 Tanh	52
0.10.1.82 TanhDrv	52
0.10.1.83 Times	53
0.10.1.84 TimesDrv	53
0.10.1.85 Transpose	53
0.10.1.86 TransposeDrv	53

0.10.1.87 Uminus . . . . .	54
0.10.1.88 UminusDrv . . . . .	54

## 0.1 GPUmat Struct Reference

### Public Attributes

- [GPUmatGPUtype](#) **gputype**
- [GPUmatNumerics](#) **numerics**
- [GPUmatFFT](#) **fft**
- [GPUmatInterface](#) \* **gmat**
- [GPUmatCompiler](#) **comp**
- struct [GPUmatModules](#) **mod**
- [GPUmatAux](#) **aux**
- [GPUmatDebug](#) **debug**

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

- GPUmat.hh

## 0.2 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*

## 0.2.1 Member Data Documentation

### 0.2.1.1 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.

### 0.2.1.2 void(\* GPUmatCompiler::createMxContext)(mxArray \*mx)

createMxContext Add an mxArray to the compilation context.

### 0.2.1.3 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

### 0.2.1.4 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

### 0.2.1.5 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

### 0.2.1.6 void(\* GPUmatCompiler::pushGPUtype)(void \*)

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

### 0.2.1.7 void(\* GPUmatCompiler::pushMx)(const mxArray \*)

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

**0.2.1.8 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

## 0.3 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*

### 0.3.1 Member Data Documentation

#### 0.3.1.1 `void(* GPUmatDebug::debugPushInstructionStack)(int)`

`debugPushInstructionStack` Pushes the instruction to the instruction stack for debugging.

#### 0.3.1.2 `int(* GPUmatDebug::getDebugMode)()`

`getDebugMode` Returns the debug mode flag.

#### 0.3.1.3 `void(* GPUmatDebug::log)(STRINGCONST char *str, int level)`

`log` Writes the specified text to the log.

#### Parameters:

← *str* Text string to be logged

#### 0.3.1.4 `void(* GPUmatDebug::logPop)()`

`logPop` Log indent pop.

**0.3.1.5 void(\* GPUmatDebug::logPush)()**

logPush Log indent push.

**0.3.1.6 void(\* GPUmatDebug::setDebugMode)(int)**

setDebugMode Set the debug mode flag.

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

- GPUmat.hh

## 0.4 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.*

### 0.4.1 Member Data Documentation

#### 0.4.1.1 GPUtype(\* GPUmatFFT::FFT1Drv)(const GPUtype &p)

FFT1D.

##### Parameters:

← *p* GPUtype input variable.

##### Returns:

A new GPUtype object

#### 0.4.1.2 GPUtype(\* GPUmatFFT::FFT2Drv)(const GPUtype &p)

FFT2D.

##### Parameters:

← *p* GPUtype input variable.

##### Returns:

A new GPUtype object

**0.4.1.3 GPUtype(\* GPUmatFFT::FFT3Drv)(const GPUtype &p)**

FFT3D.

**Parameters:**← *p* GPUtype input variable.**Returns:**

A new GPUtype object

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

IFFT1D.

**Parameters:**← *p* GPUtype input variable.**Returns:**

A new GPUtype object

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

IFFT2D.

**Parameters:**← *p* GPUtype input variable.**Returns:**

A new GPUtype object

**0.4.1.6 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

## 0.5 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 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.*

## 0.5.1 Member Data Documentation

### 0.5.1.1 `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

**0.5.1.2 GPUtype(\* GPUmatGPUtype::clone)(const GPUtype &p)**

Clones a GPUtype.

**Parameters:**

- ← *p* GPUtype to clone.

**Returns:**

The cloned GPUtype

**0.5.1.3 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.

**0.5.1.4 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

### 0.5.1.5 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

### 0.5.1.6 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)

### 0.5.1.7 GPUtype(\* GPUmatGPUtype::doubleToFloat)(const GPUtype &p)

Cast from DOUBLE to FLOAT.

#### Parameters:

- ← *p* GPUtype input variable (DOUBLE)

#### Returns:

A new GPUtype object (FLOAT)

### 0.5.1.8 void(\* GPUmatGPUtype::eye)(const GPUtype &p)

EYE. Defined in NUMERICS module

### 0.5.1.9 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

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

Cast from FLOAT to DOUBLE.

#### Parameters:

- ← *p* GPUtype input variable (FLOAT)

#### Returns:

A new GPUtype object (DOUBLE)

### 0.5.1.11 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

### 0.5.1.12 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

**0.5.1.13 GPUtype(\* GPUmatGPUtype::getGPUtype)(const mxArray \*mx)**

Creates a GPUtype from a Matlab [GPUmat](#) variable.

**Parameters:**

← *mx* [GPUmat](#) variable (GPUsingle, GPUdouble).

**Returns:**

GPUtype object

**0.5.1.14 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

**0.5.1.15 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

**0.5.1.16 const int\*(\* GPUmatGPUtype::getSize)(const GPUtype &p)**

Returns the dimensions array of a GPUtype object.

**Parameters:**

← *p* GPUtype input variable.

**Returns:**

The dimensions array

**0.5.1.17** `gpuTYPE_t(* GPUmatGPUtype::getType)(const GPUtype &p)`

Returns the type (`gpuTYPE_t`) of a `GPUtype` object.

**Parameters:**

← *p* `GPUtype` input variable.

**Returns:**

`gpuTYPE_t`

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

Returns 1 if the `GPUtype` is `COMPLEX`.

**Parameters:**

← *p* `GPUtype` input variable.

**Returns:**

1 if `COMPLEX`

**0.5.1.19** `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`

**0.5.1.20** `int(* GPUmatGPUtype::isEmpty)(const GPUtype &p)`

Returns 1 if the `GPUtype` is `EMPTY`.

**Parameters:**

← *p* `GPUtype` input variable.

**Returns:**

1 if `EMPTY`

**0.5.1.21** `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

**0.5.1.22** `int(* GPUmatGPUtype::isScalar)(const GPUtype &p)`

Returns 1 if the GPUtype is SCALAR.

**Parameters:**

← *p* GPUtype input variable.

**Returns:**

1 if SCALAR

**0.5.1.23** `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

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

MXCOLON. Wrapper to the colon function Defined in NUMERICS module

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

MXEYEDRV. Defined in NUMERICS module

**0.5.1.26 void(\* GPUmatGPUtype::mxFill)(const GPUtype &p, int nrhs, const mxArray \*prhs[ ])**

MXFILL. Wrapper to the fill function Defined in NUMERICS module

**0.5.1.27 void(\* GPUmatGPUtype::mxMemCpyDtoD)(const GPUtype &dst, const GPUtype &src, int nrhs, const mxArray \*prhs[ ])**

MXMEMCPYDTOD. Wrapper to the memCpyDtoD function Defined in NUMERICS module

**0.5.1.28 void(\* GPUmatGPUtype::mxMemCpyHtoD)(const GPUtype &dst, int nrhs, const mxArray \*prhs[ ])**

MXMEMCPYHTOD. Wrapper to the memCpyHtoD function Defined in NUMERICS module

**0.5.1.29 GPUtype(\* GPUmatGPUtype::mxOnesDrv)(const GPUtype &p, int nrhs, const mxArray \*prhs[ ])**

MXONESDRV. Defined in NUMERICS module

**0.5.1.30 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

**0.5.1.31 GPUtype(\* GPUmatGPUtype::mxPermuteDrv)(const GPUtype &RHS, int nrhs, const mxArray \*prhs[ ])**

mxPermuteDrv Defined in NUMERICS module

**0.5.1.32 GPUtype(\* GPUmatGPUtype::mxRepmatDrv)(const GPUtype &p, int nrhs, const mxArray \*prhs[ ])**

MXREPMAT. Defined in NUMERICS module

**0.5.1.33 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

**0.5.1.34 GPUtype(\* GPUmatGPUtype::mxToGPUtype)(const mxArray \*mx)**

Creates a GPUtype from a Matlab array.

**Parameters:**

- ← *mx* Matlab array.

**Returns:**

GPUtype

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

MXZEROSDRV. Defined in NUMERICS module

**0.5.1.36 void(\* GPUmatGPUtype::ones)(const GPUtype &p)**

ONES. Defined in NUMERICS module

**0.5.1.37 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

### 0.5.1.38 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

### 0.5.1.39 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)

### 0.5.1.40 GPUtype(\* GPUmatGPUtype::realToComplex)(const GPUtype &p)

Cast from REAL to COMPLEX.

#### Parameters:

- ← *p* GPUtype input variable (REAL)

#### Returns:

A new GPUtype object (COMPLEX)

### 0.5.1.41 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

**0.5.1.42 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

**0.5.1.43 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)

**0.5.1.44 void(\* GPUmatGPUtype::zeros)(const GPUtype &p)**

ZEROS. Defined in NUMERICS module

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

- GPUmat.hh

## 0.6 GPUmatInterface Struct Reference

### Public Attributes

- struct [GPUmatInterfaceFunction](#) **fun**
- struct [GPUmatInterfaceConfig](#) **config**

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

- GPUmat.hh

## 0.7 GPUmatInterfaceConfig Struct Reference

### Public Attributes

- void(\* [getMajorMinor](#) )(int \*major, int \*minor)  
*getMajorMinor*

### 0.7.1 Member Data Documentation

#### 0.7.1.1 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

## 0.8 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 index)`  
*getFunctionByNumber*

### 0.8.1 Member Data Documentation

#### 0.8.1.1 void>(\* GPUmatInterfaceFunction::getFunctionByName)(STRINGCONST char \*name)

getFunctionByName Returns the function pointer

##### Parameters:

← *name* Function name.

##### Returns:

Function pointer

#### 0.8.1.2 void>(\* GPUmatInterfaceFunction::getFunctionByNumber)(int index)

getFunctionByNumber Returns the function pointer

##### Parameters:

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

##### Returns:

Function pointer

#### 0.8.1.3 int(\* GPUmatInterfaceFunction::getFunctionNumber)(STRINGCONST char \*name)

getFunctionNumber Returns the function index

##### Parameters:

← *name* Function name.

##### Returns:

Function index

**0.8.1.4** `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

## 0.9 GPUmatModules Struct Reference

### Public Attributes

- int [gpumat](#)  
*native GPUmat functions loaded*
- int [modules](#)  
*main module*
- int [numerics](#)  
*numerics module*

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

- GPUmat.hh

## 0.10 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.*

## 0.10.1 Member Data Documentation

### 0.10.1.1 `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

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

AbsDrv function.

#### Parameters:

- ← *p* GPUtype input variable.

#### Returns:

GPUtype pointer

**0.10.1.3 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

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

AcosDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.5 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

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

AcoshDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.7 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

**0.10.1.8 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

**0.10.1.9 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

**0.10.1.10 GPUtype(\* GPUmatNumerics::AsinDrv)(const GPUtype &p)**

AsinDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.11 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

**0.10.1.12 GPUtype(\* GPUmatNumerics::AsinhDrv)(const GPUtype &p)**

AsinhDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.13 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

**0.10.1.14 GPUtype(\* GPUmatNumerics::AtanDrv)(const GPUtype &p)**

AtanDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.15 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

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

AtanhDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.17 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

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

CeilDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.19 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

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

ConjDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.21 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

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

CosDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.23 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

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

CoshDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.25 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

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

CtransposeDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.27 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

**0.10.1.28 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

**0.10.1.29 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

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

ExpDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.31 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

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

FloorDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.33 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

**0.10.1.34 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

**0.10.1.35** 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

**0.10.1.36** 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

**0.10.1.37** 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

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

ImagDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.39 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

**0.10.1.40 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

**0.10.1.41 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

**0.10.1.42 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

**0.10.1.43 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

**0.10.1.44 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

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

Log10Drv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.46 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

**0.10.1.47 GPUtype(\* GPUMatNumerics::Log1pDrv)(const GPUtype &p)**

Log1pDrv function.

**Parameters:**

←  $p$  GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.48 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

**0.10.1.49 GPUtype(\* GPUMatNumerics::Log2Drv)(const GPUtype &p)**

Log2Drv function.

**Parameters:**

←  $p$  GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.50 GPUtype(\* GPUMatNumerics::LogDrv)(const GPUtype &p)**

LogDrv function.

**Parameters:**

←  $p$  GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.51** 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

**0.10.1.52** 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

**0.10.1.53** 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

**0.10.1.54** 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

**0.10.1.55** `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

**0.10.1.56** `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

**0.10.1.57** `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

**0.10.1.58** `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

**0.10.1.59** 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

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

NotDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.61** 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

**0.10.1.62** 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

**0.10.1.63 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

**0.10.1.64 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

**0.10.1.65 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

**0.10.1.66 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

**0.10.1.67 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

**0.10.1.68 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

**0.10.1.69 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

**0.10.1.70 GPUtype(\* GPUmatNumerics::RealDrv)(const GPUtype &p)**

RealDrv function.

**Parameters:**

- ←  $p$  GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.71 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

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

RoundDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.73 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

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

SinDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.75 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

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

SinhDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.77 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

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

SqrtDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.79 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

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

TanDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.81 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

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

TanhDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.83 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

**0.10.1.84 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

**0.10.1.85 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

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

TransposeDrv function.

**Parameters:**

- ← *p* GPUtype input variable.

**Returns:**

GPUtype pointer

**0.10.1.87 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

**0.10.1.88 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

# Index

- abort
  - GPUmatCompiler, 3
- Abs
  - GPUmatNumerics, 32
- AbsDrv
  - GPUmatNumerics, 32
- Acos
  - GPUmatNumerics, 32
- AcosDrv
  - GPUmatNumerics, 33
- Acosh
  - GPUmatNumerics, 33
- AcoshDrv
  - GPUmatNumerics, 33
- And
  - GPUmatNumerics, 33
- AndDrv
  - GPUmatNumerics, 34
- Asin
  - GPUmatNumerics, 34
- AsinDrv
  - GPUmatNumerics, 34
- Asinh
  - GPUmatNumerics, 34
- AsinhDrv
  - GPUmatNumerics, 35
- assign
  - GPUmatGPUtype, 11
- Atan
  - GPUmatNumerics, 35
- AtanDrv
  - GPUmatNumerics, 35
- Atanh
  - GPUmatNumerics, 35
- AtanhDrv
  - GPUmatNumerics, 36
- Ceil
  - GPUmatNumerics, 36
- CeilDrv
  - GPUmatNumerics, 36
- clone
  - GPUmatGPUtype, 12
- colon
  - GPUmatGPUtype, 12
- Conj
  - GPUmatNumerics, 36
- ConjDrv
  - GPUmatNumerics, 37
- Cos
  - GPUmatNumerics, 37
- CosDrv
  - GPUmatNumerics, 37
- Cosh
  - GPUmatNumerics, 37
- CoshDrv
  - GPUmatNumerics, 38
- create
  - GPUmatGPUtype, 12
- createMx
  - GPUmatGPUtype, 12
- createMxArray
  - GPUmatGPUtype, 13
- createMxContext
  - GPUmatCompiler, 3
- Ctranspose
  - GPUmatNumerics, 38
- CtransposeDrv
  - GPUmatNumerics, 38
- debugPushInstructionStack
  - GPUmatDebug, 5
- doubleToFloat
  - GPUmatGPUtype, 13
- Eq
  - GPUmatNumerics, 38
- EqDrv
  - GPUmatNumerics, 39
- Exp
  - GPUmatNumerics, 39
- ExpDrv
  - GPUmatNumerics, 39
- eye
  - GPUmatGPUtype, 13
- FFT1Drv
  - GPUmatFFT, 7
- FFT2Drv
  - GPUmatFFT, 7

- FFT3Drv
  - GPUmatFFT, 7
- fill
  - GPUmatGPUtype, 13
- floatToDouble
  - GPUmatGPUtype, 14
- Floor
  - GPUmatNumerics, 39
- FloorDrv
  - GPUmatNumerics, 40
- Ge
  - GPUmatNumerics, 40
- GeDrv
  - GPUmatNumerics, 40
- getCompileMode
  - GPUmatCompiler, 3
- getContextGPUtype
  - GPUmatCompiler, 3
- getContextMx
  - GPUmatCompiler, 3
- getDataSize
  - GPUmatGPUtype, 14
- getDebugMode
  - GPUmatDebug, 5
- getFunctionByName
  - GPUmatInterfaceFunction, 24
- getFunctionByNumber
  - GPUmatInterfaceFunction, 24
- getFunctionNumber
  - GPUmatInterfaceFunction, 24
- getGPUptr
  - GPUmatGPUtype, 14
- getGPUtype
  - GPUmatGPUtype, 14
- getMajorMinor
  - GPUmatInterfaceConfig, 23
- getNdims
  - GPUmatGPUtype, 15
- getNumel
  - GPUmatGPUtype, 15
- getSize
  - GPUmatGPUtype, 15
- getType
  - GPUmatGPUtype, 15
- GPUmat, 1
- GPUmatCompiler, 2
  - abort, 3
  - createMxContext, 3
  - getCompileMode, 3
  - getContextGPUtype, 3
  - getContextMx, 3
  - pushGPUtype, 3
  - pushMx, 3
  - registerInstruction, 3
- GPUmatDebug, 5
  - debugPushInstructionStack, 5
  - getDebugMode, 5
  - log, 5
  - logPop, 5
  - logPush, 5
  - setDebugMode, 6
- GPUmatFFT, 7
  - FFT1Drv, 7
  - FFT2Drv, 7
  - FFT3Drv, 7
  - IFFT1Drv, 8
  - IFFT2Drv, 8
  - IFFT3Drv, 8
- GPUmatGPUtype, 9
  - assign, 11
  - clone, 12
  - colon, 12
  - create, 12
  - createMx, 12
  - createMxArray, 13
  - doubleToFloat, 13
  - eye, 13
  - fill, 13
  - floatToDouble, 14
  - getDataSize, 14
  - getGPUptr, 14
  - getGPUtype, 14
  - getNdims, 15
  - getNumel, 15
  - getSize, 15
  - getType, 15
  - isComplex, 16
  - isDouble, 16
  - isEmpty, 16
  - isFloat, 16
  - isScalar, 17
  - mxAssign, 17
  - mxColonDrv, 17
  - mxEyeDrv, 17
  - mxFill, 17
  - mxMemCpyDtoD, 18
  - mxMemCpyHtoD, 18
  - mxOnesDrv, 18
  - mxPermute, 18
  - mxPermuteDrv, 18
  - mxRepmatDrv, 18
  - mxSlice, 18
  - mxToGPUtype, 19
  - mxZerosDrv, 19
  - ones, 19
  - permute, 19
  - realimag, 19

- realImagToComplex, 20
- realToComplex, 20
- setSize, 20
- slice, 20
- toMxArray, 21
- zeros, 21
- GPUmatInterface, 22
- GPUmatInterfaceConfig, 23
  - getMajorMinor, 23
- GPUmatInterfaceFunction, 24
  - getFunctionByName, 24
  - getFunctionByNumber, 24
  - getFunctionNumber, 24
  - registerFunction, 24
- GPUmatModules, 26
- GPUmatNumerics, 27
  - Abs, 32
  - AbsDrv, 32
  - Acos, 32
  - AcosDrv, 33
  - Acosh, 33
  - AcoshDrv, 33
  - And, 33
  - AndDrv, 34
  - Asin, 34
  - AsinDrv, 34
  - Asinh, 34
  - AsinhDrv, 35
  - Atan, 35
  - AtanDrv, 35
  - Atanh, 35
  - AtanhDrv, 36
  - Ceil, 36
  - CeilDrv, 36
  - Conj, 36
  - ConjDrv, 37
  - Cos, 37
  - CosDrv, 37
  - Cosh, 37
  - CoshDrv, 38
  - Ctranspose, 38
  - CtransposeDrv, 38
  - Eq, 38
  - EqDrv, 39
  - Exp, 39
  - ExpDrv, 39
  - Floor, 39
  - FloorDrv, 40
  - Ge, 40
  - GeDrv, 40
  - Gt, 40
  - GtDrv, 41
  - Imag, 41
  - ImagDrv, 41
  - Ldivide, 41
  - LdivideDrv, 42
  - Le, 42
  - LeDrv, 42
  - Log, 42
  - Log10, 43
  - Log10Drv, 43
  - Log1p, 43
  - Log1pDrv, 43
  - Log2, 44
  - Log2Drv, 44
  - LogDrv, 44
  - Lt, 44
  - LtDrv, 45
  - Minus, 45
  - MinusDrv, 45
  - Mtimes, 45
  - MtimesDrv, 46
  - Ne, 46
  - NeDrv, 46
  - Not, 46
  - NotDrv, 47
  - Or, 47
  - OrDrv, 47
  - Plus, 47
  - PlusDrv, 48
  - Power, 48
  - PowerDrv, 48
  - Rdivide, 48
  - RdivideDrv, 49
  - Real, 49
  - RealDrv, 49
  - Round, 49
  - RoundDrv, 50
  - Sin, 50
  - SinDrv, 50
  - Sinh, 50
  - SinhDrv, 51
  - Sqrt, 51
  - SqrtDrv, 51
  - Tan, 51
  - TanDrv, 52
  - Tanh, 52
  - TanhDrv, 52
  - Times, 52
  - TimesDrv, 53
  - Transpose, 53
  - TransposeDrv, 53
  - Uminus, 53
  - UminusDrv, 54
- Gt
  - GPUmatNumerics, 40
- GtDrv
  - GPUmatNumerics, 41

- IFFT1Drv
  - GPUmatFFT, 8
- IFFT2Drv
  - GPUmatFFT, 8
- IFFT3Drv
  - GPUmatFFT, 8
- Imag
  - GPUmatNumerics, 41
- ImagDrv
  - GPUmatNumerics, 41
- isComplex
  - GPUmatGPUtype, 16
- isDouble
  - GPUmatGPUtype, 16
- isEmpty
  - GPUmatGPUtype, 16
- isFloat
  - GPUmatGPUtype, 16
- isScalar
  - GPUmatGPUtype, 17
  
- Ldivide
  - GPUmatNumerics, 41
- LdivideDrv
  - GPUmatNumerics, 42
- Le
  - GPUmatNumerics, 42
- LeDrv
  - GPUmatNumerics, 42
- Log
  - GPUmatNumerics, 42
- log
  - GPUmatDebug, 5
- Log10
  - GPUmatNumerics, 43
- Log10Drv
  - GPUmatNumerics, 43
- Log1p
  - GPUmatNumerics, 43
- Log1pDrv
  - GPUmatNumerics, 43
- Log2
  - GPUmatNumerics, 44
- Log2Drv
  - GPUmatNumerics, 44
- LogDrv
  - GPUmatNumerics, 44
- logPop
  - GPUmatDebug, 5
- logPush
  - GPUmatDebug, 5
- Lt
  - GPUmatNumerics, 44
- LtDrv
  - GPUmatNumerics, 45
  
- Minus
  - GPUmatNumerics, 45
- MinusDrv
  - GPUmatNumerics, 45
- Mtimes
  - GPUmatNumerics, 45
- MtimesDrv
  - GPUmatNumerics, 46
- mxAssign
  - GPUmatGPUtype, 17
- mxColonDrv
  - GPUmatGPUtype, 17
- mxEyeDrv
  - GPUmatGPUtype, 17
- mxFill
  - GPUmatGPUtype, 17
- mxMemCpyDtoD
  - GPUmatGPUtype, 18
- mxMemCpyHtoD
  - GPUmatGPUtype, 18
- mxOnesDrv
  - GPUmatGPUtype, 18
- mxPermute
  - GPUmatGPUtype, 18
- mxPermuteDrv
  - GPUmatGPUtype, 18
- mxRepmatDrv
  - GPUmatGPUtype, 18
- mxSlice
  - GPUmatGPUtype, 18
- mxToGPUtype
  - GPUmatGPUtype, 19
- mxZerosDrv
  - GPUmatGPUtype, 19
  
- Ne
  - GPUmatNumerics, 46
- NeDrv
  - GPUmatNumerics, 46
- Not
  - GPUmatNumerics, 46
- NotDrv
  - GPUmatNumerics, 47
  
- ones
  - GPUmatGPUtype, 19
- Or
  - GPUmatNumerics, 47
- OrDrv
  - GPUmatNumerics, 47
  
- permute

- GPUmatGPUtype, 19
- Plus
  - GPUmatNumerics, 47
- PlusDrv
  - GPUmatNumerics, 48
- Power
  - GPUmatNumerics, 48
- PowerDrv
  - GPUmatNumerics, 48
- pushGPUtype
  - GPUmatCompiler, 3
- pushMx
  - GPUmatCompiler, 3
- Rdivide
  - GPUmatNumerics, 48
- RdivideDrv
  - GPUmatNumerics, 49
- Real
  - GPUmatNumerics, 49
- RealDrv
  - GPUmatNumerics, 49
- realimag
  - GPUmatGPUtype, 19
- realImagToComplex
  - GPUmatGPUtype, 20
- realToComplex
  - GPUmatGPUtype, 20
- registerFunction
  - GPUmatInterfaceFunction, 24
- registerInstruction
  - GPUmatCompiler, 3
- Round
  - GPUmatNumerics, 49
- RoundDrv
  - GPUmatNumerics, 50
- setDebugMode
  - GPUmatDebug, 6
- setSize
  - GPUmatGPUtype, 20
- Sin
  - GPUmatNumerics, 50
- SinDrv
  - GPUmatNumerics, 50
- Sinh
  - GPUmatNumerics, 50
- SinhDrv
  - GPUmatNumerics, 51
- slice
  - GPUmatGPUtype, 20
- Sqrt
  - GPUmatNumerics, 51
- SqrtDrv
  - GPUmatNumerics, 51
- Tan
  - GPUmatNumerics, 51
- TanDrv
  - GPUmatNumerics, 52
- Tanh
  - GPUmatNumerics, 52
- TanhDrv
  - GPUmatNumerics, 52
- Times
  - GPUmatNumerics, 52
- TimesDrv
  - GPUmatNumerics, 53
- toMxArray
  - GPUmatGPUtype, 21
- Transpose
  - GPUmatNumerics, 53
- TransposeDrv
  - GPUmatNumerics, 53
- Uminus
  - GPUmatNumerics, 53
- UminusDrv
  - GPUmatNumerics, 54
- zeros
  - GPUmatGPUtype, 21