

LIBMATIO API 1.3.3

Christopher Hulbert

14 Dec 2008

Contents

1	Library Documentation	3
1.1	Matlab MAT File I/O Library	3
1.2	Internal Functions	21
2	Data Structure Documentation	53
2.1	ComplexSplit Struct Reference	53
2.2	mat_t Struct Reference	54
2.3	matvar_t Struct Reference	56
2.4	sparse_t Struct Reference	60

Chapter 1

Library Documentation

1.1 Matlab MAT File I/O Library

Data Structures

- struct [ComplexSplit](#)
Complex data type using split storage.
- struct [mat_t](#)
Matlab MAT File information.
- struct [matvar_t](#)
Matlab variable information.
- struct [sparse_t](#)
sparse data information

Enumerations

- enum { [BY_NAME](#) = 1, [BY_INDEX](#) = 2 }
matio lookup type
- enum [mat_acc](#) { [MAT_ACC_RDONLY](#) = 1, [MAT_ACC_RDWR](#) = 2 }
MAT file access types.
- enum [mat_ft](#) { [MAT_FT_MAT5](#) = 1, [MAT_FT_MAT4](#) = 1 << 16 }
MAT file versions.
- enum [matio_classes](#) {
 [MAT_C_CELL](#) = 1, [MAT_C_STRUCT](#) = 2, [MAT_C_OBJECT](#) = 3, [MAT_C_CHAR](#) = 4,
 [MAT_C_SPARSE](#) = 5, [MAT_C_DOUBLE](#) = 6, [MAT_C_SINGLE](#) = 7, [MAT_C_INT8](#) = 8,
 [MAT_C_UINT8](#) = 9, [MAT_C_INT16](#) = 10, [MAT_C_UINT16](#) = 11, [MAT_C_INT32](#) = 12,
 [MAT_C_UINT32](#) = 13, [MAT_C_INT64](#) = 14, [MAT_C_UINT64](#) = 15, [MAT_C_FUNCTION](#) = 16
}

Matlab variable classes.

- enum `matio_compression` { `COMPRESSION_NONE` = 0, `COMPRESSION_ZLIB` = 1 }

Matlab compression options.

- enum `matio_flags` { `MAT_F_COMPLEX` = 0x0800, `MAT_F_GLOBAL` = 0x0400, `MAT_F_LOGICAL` = 0x0200, `MAT_F_CLASS_T` = 0x00ff }

Matlab array flags.

- enum `matio_types` {
`MAT_T_UNKNOWN` = 0, `MAT_T_INT8` = 1, `MAT_T_UINT8` = 2, `MAT_T_INT16` = 3,
`MAT_T_UINT16` = 4, `MAT_T_INT32` = 5, `MAT_T_UINT32` = 6, `MAT_T_SINGLE` = 7,
`MAT_T_DOUBLE` = 9, `MAT_T_INT64` = 12, `MAT_T_UINT64` = 13, `MAT_T_MATRIX` = 14,
`MAT_T_COMPRESSED` = 15, `MAT_T_UTF8` = 16, `MAT_T_UTF16` = 17, `MAT_T_UTF32` = 18,
`MAT_T_STRING` = 20, `MAT_T_CELL` = 21, `MAT_T_STRUCT` = 22, `MAT_T_ARRAY` = 23,
`MAT_T_FUNCTION` = 24 }

Matlab data types.

Functions

- int `Mat_CalcSingleSubscript` (int rank, int *dims, int *subs)
Calculate a single subscript from a set of subscript values.
- int * `Mat_CalcSubscripts` (int rank, int *dims, int index)
Calculate a set of subscript values from a single(linear) subscript.
- int `Mat_Close` (`mat_t` *mat)
Closes an open Matlab MAT file.
- `mat_t` * `Mat_Create` (const char *matname, const char *hdr_str)
Creates a new Matlab MAT file.
- `mat_t` * `Mat_Open` (const char *matname, int mode)
Opens an existing Matlab MAT file.
- int `Mat_Rewind` (`mat_t` *mat)
Rewinds a Matlab MAT file to the first variable.
- size_t `Mat_SizeOfClass` (int class_type)
Returns the size of a Matlab Class.
- int `Mat_VarAddStructField` (`matvar_t` *matvar, `matvar_t` **fields)
Adds a field to a structure.
- `matvar_t` * `Mat_VarCalloc` (void)
Allocates memory for a new `matvar_t` and initializes all the fields.

- `matvar_t * Mat_VarCreate` (`const char *name`, `int class_type`, `int data_type`, `int rank`, `int *dims`, `void *data`, `int opt`)
Creates a MAT Variable with the given name and (optionally) data.
- `int Mat_VarDelete` (`mat_t *mat`, `char *name`)
Deletes a variable from a file.
- `matvar_t * Mat_VarDuplicate` (`const matvar_t *in`, `int opt`)
Duplicates a `matvar_t` structure.
- `void Mat_VarFree` (`matvar_t *matvar`)
Frees all the allocated memory associated with the structure.
- `matvar_t * Mat_VarGetCell` (`matvar_t *matvar`, `int index`)
Returns a pointer to the Cell array at a specific index.
- `matvar_t ** Mat_VarGetCells` (`matvar_t *matvar`, `int *start`, `int *stride`, `int *edge`)
Indexes a cell array.
- `matvar_t ** Mat_VarGetCellsLinear` (`matvar_t *matvar`, `int start`, `int stride`, `int edge`)
Indexes a cell array.
- `int Mat_VarGetNumberOfFields` (`matvar_t *matvar`)
Returns the number of fields in a structure variable.
- `size_t Mat_VarGetSize` (`matvar_t *matvar`)
Calculates the size of a matlab variable in bytes.
- `matvar_t * Mat_VarGetStructField` (`matvar_t *matvar`, `void *name_or_index`, `int opt`, `int index`)
Finds a field of a structure.
- `matvar_t * Mat_VarGetStructs` (`matvar_t *matvar`, `int *start`, `int *stride`, `int *edge`, `int copy_fields`)
Indexes a structure.
- `matvar_t * Mat_VarGetStructsLinear` (`matvar_t *matvar`, `int start`, `int stride`, `int edge`, `int copy_fields`)
Indexes a structure.
- `void Mat_VarPrint` (`matvar_t *matvar`, `int printdata`)
Prints the variable information.
- `matvar_t * Mat_VarRead` (`mat_t *mat`, `char *name`)
Reads the variable with the given name from a MAT file.
- `int Mat_VarReadData` (`mat_t *mat`, `matvar_t *matvar`, `void *data`, `int *start`, `int *stride`, `int *edge`)
Reads MAT variable data from a file.
- `int Mat_VarReadDataAll` (`mat_t *mat`, `matvar_t *matvar`)
Reads all the data for a matlab variable.

- `int Mat_VarReadDataLinear (mat_t *mat, matvar_t *matvar, void *data, int start, int stride, int edge)`
Reads MAT variable data from a file.
- `matvar_t * Mat_VarReadInfo (mat_t *mat, char *name)`
Reads the information of a variable with the given name from a MAT file.
- `matvar_t * Mat_VarReadNext (mat_t *mat)`
Reads the next variable in a MAT file.
- `matvar_t * Mat_VarReadNextInfo (mat_t *mat)`
Reads the information of the next variable in a MAT file.
- `int Mat_VarWrite (mat_t *mat, matvar_t *matvar, int compress)`
Writes the given MAT variable to a MAT file.
- `int Mat_VarWriteData (mat_t *mat, matvar_t *matvar, void *data, int *start, int *stride, int *edge)`
Writes the given data to the MAT variable.
- `int Mat_VarWriteInfo (mat_t *mat, matvar_t *matvar)`
Writes the given MAT variable to a MAT file.

1.1.1 Detailed Description

1.1.2 Enumeration Type Documentation

1.1.2.1 anonymous enum

matio lookup type

Enumerator:

BY_NAME Lookup by name
BY_INDEX Lookup by index

1.1.2.2 enum mat_acc

MAT file access types

Enumerator:

MAT_ACC_RDONLY Read only file access.
MAT_ACC_RDWR Read/Write file access.

1.1.2.3 enum mat_ft

MAT file versions

Enumerator:

MAT_FT_MAT5 Matlab level-5 file.

MAT_FT_MAT4 Version 4 file.

1.1.2.4 enum matio_classes

Matlab variable classes

Enumerator:

MAT_C_CELL Matlab cell array class.

MAT_C_STRUCT Matlab structure class.

MAT_C_OBJECT Matlab object class.

MAT_C_CHAR Matlab character array class.

MAT_C_SPARSE Matlab sparse array class.

MAT_C_DOUBLE Matlab double-precision class.

MAT_C_SINGLE Matlab single-precision class.

MAT_C_INT8 Matlab signed 8-bit integer class.

MAT_C_UINT8 Matlab unsigned 8-bit integer class.

MAT_C_INT16 Matlab signed 16-bit integer class.

MAT_C_UINT16 Matlab unsigned 16-bit integer class.

MAT_C_INT32 Matlab signed 32-bit integer class.

MAT_C_UINT32 Matlab unsigned 32-bit integer class.

MAT_C_INT64 Matlab unsigned 32-bit integer class.

MAT_C_UINT64 Matlab unsigned 32-bit integer class.

MAT_C_FUNCTION Matlab unsigned 32-bit integer class.

1.1.2.5 enum matio_compression

Matlab compression options

Enumerator:

COMPRESSION_NONE No compression.

COMPRESSION_ZLIB zlib compression

1.1.2.6 enum matio_flags

Matlab array flags

Enumerator:

MAT_F_COMPLEX Complex bit flag.
MAT_F_GLOBAL Global bit flag.
MAT_F_LOGICAL Logical bit flag.
MAT_F_CLASS_T Class-Type bits flag.

1.1.2.7 enum matio_types

Matlab data types

Enumerator:

MAT_T_UNKNOWN UNKOWN data type.
MAT_T_INT8 8-bit signed integer data type
MAT_T_UINT8 8-bit unsigned integer data type
MAT_T_INT16 16-bit signed integer data type
MAT_T_UINT16 16-bit unsigned integer data type
MAT_T_INT32 32-bit signed integer data type
MAT_T_UINT32 32-bit unsigned integer data type
MAT_T_SINGLE IEEE 754 single precision data type.
MAT_T_DOUBLE IEEE 754 double precision data type.
MAT_T_INT64 64-bit signed integer data type
MAT_T_UINT64 64-bit unsigned integer data type
MAT_T_MATRIX matrix data type
MAT_T_COMPRESSED compressed data type
MAT_T_UTF8 8-bit unicode text data type
MAT_T_UTF16 16-bit unicode text data type
MAT_T_UTF32 32-bit unicode text data type
MAT_T_STRING String data type.
MAT_T_CELL Cell array data type.
MAT_T_STRUCT Structure data type.
MAT_T_ARRAY Array data type.
MAT_T_FUNCTION Function data type.

1.1.3 Function Documentation

1.1.3.1 int Mat_CalcSingleSubscript (int rank, int * dims, int * subs)

Calculates a single linear subscript (0-relative) given a 1-relative subscript for each dimension. The calculation uses the formula below where index is the linear index, s is an array of length RANK where each

element is the subscript for the corresponding dimension, D is an array whose elements are the dimensions of the variable.

$$index = \sum_{k=0}^{RANK-1} [(s_k - 1) \prod_{l=0}^k D_l]$$

Parameters:

rank Rank of the variable

dims dimensions of the variable

subs Dimension subscripts

Returns:

Single (linear) subscript

1.1.3.2 int* Mat_CalcSubscripts (int rank, int * dims, int index)

Calculates 1-relative subscripts for each dimension given a 0-relative linear index. Subscripts are calculated as follows where s is the array of dimension subscripts, D is the array of dimensions, and index is the linear index.

$$s_k = \lfloor \frac{1}{L} \prod_{l=0}^k D_l \rfloor + 1$$

$$L = index - \sum_{l=k}^{RANK-1} s_l \prod_{m=0}^l D_m$$

Parameters:

rank Rank of the variable

dims dimensions of the variable

index linear index

Returns:

Array of dimension subscripts

1.1.3.3 int Mat_Close (mat_t * mat)

Closes the given Matlab MAT file and frees any memory with it.

Parameters:

mat Pointer to the MAT file

Return values:

0

References mat_t::filename, mat_t::fp, mat_t::header, and mat_t::subsys_offset.

Referenced by Mat_Open(), and Mat_VarDelete().

1.1.3.4 `mat_t* Mat_Create (const char * matname, const char * hdr_str)`

Tries to create a new Matlab MAT file with the given name and optional header string. If no header string is given, the default string is used containing the software, version, and date in it. If a header string is given, at most the first 116 characters is written to the file. The given header string need not be the full 116 characters, but MUST be NULL terminated.

Parameters:

matname Name of MAT file to create

hdr_str Optional header string, NULL to use default

Returns:

A pointer to the MAT file or NULL if it failed. This is not a simple FILE * and should not be used as one.

References `mat_t::bof`, `mat_t::byteswap`, `mat_t::filename`, `mat_t::fp`, `mat_t::header`, `MAT_ACC_RDWR`, `mat_t::mode`, `mat_t::subsys_offset`, and `mat_t::version`.

Referenced by `Mat_Open()`, and `Mat_VarDelete()`.

1.1.3.5 `mat_t* Mat_Open (const char * matname, int mode)`

Tries to open a Matlab MAT file with the given name

Parameters:

matname Name of MAT file to open

mode File access mode (`MAT_ACC_RDONLY`, `MAT_ACC_RDWR`, etc).

Returns:

A pointer to the MAT file or NULL if it failed. This is not a simple FILE * and should not be used as one.

References `mat_t::bof`, `mat_t::byteswap`, `mat_t::filename`, `mat_t::fp`, `mat_t::header`, `MAT_ACC_RDONLY`, `MAT_ACC_RDWR`, `Mat_Close()`, `Mat_Create()`, `MAT_FT_MAT4`, `Mat_int16Swap()`, `mat_t::mode`, `mat_t::subsys_offset`, and `mat_t::version`.

Referenced by `Mat_VarDelete()`.

1.1.3.6 `int Mat_Rewind (mat_t * mat)`

Rewinds a Matlab MAT file to the first variable

Parameters:

mat Pointer to the MAT file

Return values:

0 on success

References `mat_t::fp`, `MAT_FT_MAT4`, and `mat_t::version`.

1.1.3.7 `size_t Mat_SizeOfClass (int class_type)`

Returns the size (in bytes) of the matlab class `class_type`

Parameters:

class_type Matlab class type (MAT_C_*)

Returns:

Size of the class

References MAT_C_CHAR, MAT_C_DOUBLE, MAT_C_INT16, MAT_C_INT32, MAT_C_INT64, MAT_C_INT8, MAT_C_SINGLE, MAT_C_UINT16, MAT_C_UINT32, MAT_C_UINT64, and MAT_C_UINT8.

Referenced by `Mat_VarGetSize()`.

1.1.3.8 `int Mat_VarAddStructField (matvar_t * matvar, matvar_t **fields)`

Adds the given field to the structure. `fields` should be an array of `matvar_t` pointers of the same size as the structure (i.e. 1 field per structure element).

Parameters:

matvar Pointer to the Structure MAT variable

fields Array of fields to be added

Return values:

0 on success

References `matvar_t::data`, `matvar_t::dims`, `matvar_t::nbytes`, and `matvar_t::rank`.

1.1.3.9 `matvar_t* Mat_VarCalloc (void)`**Returns:**

A newly allocated `matvar_t`

References `matvar_t::class_type`, `matvar_t::compression`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `matvar_t::fp`, `matvar_t::fpos`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `matvar_t::mem_conserve`, `matvar_t::name`, `matvar_t::nbytes`, `matvar_t::rank`, and `matvar_t::z`.

Referenced by `Mat_VarCreate()`, `Mat_VarReadNextInfo5()`, and `ReadNextCell()`.

1.1.3.10 `matvar_t* Mat_VarCreate (const char * name, int class_type, int data_type, int rank, int * dims, void * data, int opt)`

Creates a MAT variable that can be written to a Matlab MAT file with the given name, data type, dimensions and data. Rank should always be 2 or more. i.e. Scalar values would have rank=2 and `dims[2] = { 1,1 }`. Data type is one of the MAT_T types. MAT adds MAT_T_STRUCT and MAT_T_CELL to create Structures and Cell Arrays respectively. For MAT_T_STRUCT, data should be a NULL terminated array of `matvar_t`

* variables (i.e. for a 3x2 structure with 10 fields, there should be 61 `matvar_t` * variables where the last one is NULL). For cell arrays, the NULL termination isn't necessary. So to create a cell array of size 3x2, data would be the address of an array of 6 `matvar_t` * variables.

EXAMPLE: To create a struct of size 3x2 with 3 fields:

```
int rank=2, dims[2] = {3,2}, nfields = 3;
matvar_t **vars;

vars = malloc((3*2*nfields+1)*sizeof(matvar_t *));
vars[0] = Mat_VarCreate(...);
:
vars[3*2*nfields-1] = Mat_VarCreate(...);
vars[3*2*nfields] = NULL;
```

EXAMPLE: To create a cell array of size 3x2:

```
int rank=2, dims[2] = {3,2};
matvar_t **vars;

vars = malloc(3*2*sizeof(matvar_t *));
vars[0] = Mat_VarCreate(...);
:
vars[5] = Mat_VarCreate(...);
```

Parameters:

name Name of the variable to create

class_type class type of the variable in Matlab(one of the mx Classes)

data_type data type of the variable (one of the MAT_T_ Types)

rank Rank of the variable

dims array of dimensions of the variable of size rank

data pointer to the data

opt 0, or bitwise or of the following options:

- MEM_CONSERVE to just use the pointer to the data and not copy the data itself. Note that the pointer should not be freed until you are done with the mat variable. The `Mat_VarFree` function will NOT free data that was created with MEM_CONSERVE, so free it yourself.
- MAT_F_COMPLEX to specify that the data is complex. The data variable should be a contiguous piece of memory with the real part written first and the imaginary second
- MAT_F_GLOBAL to assign the variable as a global variable
- MAT_F_LOGICAL to specify that it is a logical variable

Returns:

A MAT variable that can be written to a file or otherwise used

References `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_NONE`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::dims`, `ComplexSplit::Im`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `MAT_C_CHAR`, `MAT_C_SPARSE`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_CELL`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_STRUCT`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `MAT_T_UTF16`, `MAT_T_UTF32`, `MAT_T_UTF8`, `Mat_VarCalloc()`, `Mat_VarFree()`, `matvar_t::mem_conserve`, `matvar_t::name`, `matvar_t::nbytes`, `matvar_t::rank`, and `ComplexSplit::Re`.

1.1.3.11 int Mat_VarDelete (mat_t * *mat*, char * *name*)**Parameters:***mat* Pointer to the [mat_t](#) file structure*name* Name of the variable to delete**Returns:**

0 on success

References [mat_t::filename](#), [mat_t::fp](#), [mat_t::header](#), [Mat_Close\(\)](#), [Mat_Create\(\)](#), [Mat_Open\(\)](#), [Mat_VarFree\(\)](#), [Mat_VarReadNext\(\)](#), [Mat_VarWrite\(\)](#), [mat_t::mode](#), and [matvar_t::name](#).

1.1.3.12 matvar_t* Mat_VarDuplicate (const matvar_t * *in*, int *opt*)

Provides a clean function for duplicating a [matvar_t](#) structure.

Parameters:*in* pointer to the [matvar_t](#) structure to be duplicated

opt 0 does a shallow duplicate and only assigns the data pointer to the duplicated array. 1 will do a deep duplicate and actually duplicate the contents of the data. Warning: If you do a shallow copy and free both structures, the data will be freed twice and memory will be corrupted. This may be fixed in a later release.

Returns:

Pointer to the duplicated [matvar_t](#) structure.

References [matvar_t::class_type](#), [matvar_t::compression](#), [matvar_t::data](#), [matvar_t::data_size](#), [matvar_t::data_type](#), [matvar_t::datapos](#), [matvar_t::dims](#), [matvar_t::fpos](#), [ComplexSplit::Im](#), [matvar_t::isComplex](#), [matvar_t::isGlobal](#), [matvar_t::isLogical](#), [MAT_C_CELL](#), [MAT_C_STRUCT](#), [Mat_VarDuplicate\(\)](#), [matvar_t::mem_conserve](#), [matvar_t::name](#), [matvar_t::nbytes](#), [matvar_t::rank](#), [ComplexSplit::Re](#), and [matvar_t::z](#).

Referenced by [Mat_VarDuplicate\(\)](#), [Mat_VarGetStructs\(\)](#), and [Mat_VarGetStructsLinear\(\)](#).

1.1.3.13 void Mat_VarFree (matvar_t * *matvar*)

Frees memory used by a MAT variable. Frees the data associated with a MAT variable if it's non-NULL and MEM_CONSERVE was not used.

Parameters:*matvar* Pointer to the [matvar_t](#) structure

References [matvar_t::class_type](#), [matvar_t::compression](#), [COMPRESSION_ZLIB](#), [sparse_t::data](#), [matvar_t::data](#), [matvar_t::data_size](#), [matvar_t::dims](#), [ComplexSplit::Im](#), [sparse_t::ir](#), [matvar_t::isComplex](#), [sparse_t::jc](#), [MAT_C_CELL](#), [MAT_C_SPARSE](#), [MAT_C_STRUCT](#), [Mat_VarFree\(\)](#), [matvar_t::mem_conserve](#), [matvar_t::name](#), [matvar_t::nbytes](#), [ComplexSplit::Re](#), and [matvar_t::z](#).

Referenced by [Mat_VarCreate\(\)](#), [Mat_VarDelete\(\)](#), [Mat_VarFree\(\)](#), [Mat_VarGetStructs\(\)](#), [Mat_VarReadInfo\(\)](#), [Mat_VarReadNextInfo5\(\)](#), [ReadNextCell\(\)](#), and [ReadNextStructField\(\)](#).

1.1.3.14 `matvar_t* Mat_VarGetCell (matvar_t * matvar, int index)`

Returns a pointer to the Cell Array Field at the given 1-relative index. MAT file must be a version 5 matlab file.

Parameters:

matvar Pointer to the Cell Array MAT variable
index linear index of cell to return

Returns:

Pointer to the Cell Array Field on success, NULL on error

References `matvar_t::data`, `matvar_t::dims`, and `matvar_t::rank`.

1.1.3.15 `matvar_t Mat_VarGetCells (matvar_t * matvar, int * start, int * stride, int * edge)`**

Finds cells of a cell array given a start, stride, and edge for each dimension. The cells are placed in a pointer array. The cells should not be freed, but the array of pointers should be. If copies are needed, use `Mat_VarDuplicate` on each cell. MAT File version must be 5.

Parameters:

matvar Cell Array matlab variable
start vector of length rank with 0-relative starting coordinates for each dimension.
stride vector of length rank with strides for each dimension.
edge vector of length rank with the number of elements to read in each dimension.

Returns:

an array of pointers to the cells

References `matvar_t::data`, `matvar_t::dims`, and `matvar_t::rank`.

1.1.3.16 `matvar_t Mat_VarGetCellsLinear (matvar_t * matvar, int start, int stride, int edge)`**

Finds cells of a cell array given a linear indexed start, stride, and edge. The cells are placed in a pointer array. The cells themselves should not be freed as they are part of the original cell array, but the pointer array should be. If copies are needed, use `Mat_VarDuplicate` on each of the cells. MAT file version must be 5.

Parameters:

matvar Cell Array matlab variable
start starting index
stride stride
edge Number of cells to get

Returns:

an array of pointers to the cells

References `matvar_t::data`, and `matvar_t::rank`.

1.1.3.17 int Mat_VarGetNumberOfFields (matvar_t * matvar)

Returns the number of fields in the given structure. MAT file version must be 5.

Parameters:

matvar Structure matlab variable

Returns:

Number of fields, or a negative number on error

References `matvar_t::class_type`, `matvar_t::data_size`, `matvar_t::dims`, `MAT_C_STRUCT`, `matvar_t::nbytes`, and `matvar_t::rank`.

1.1.3.18 size_t Mat_VarGetSize (matvar_t * matvar)**Parameters:**

matvar matlab variable

Returns:

size of the variable in bytes

References `matvar_t::class_type`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::dims`, `MAT_C_CELL`, `MAT_C_STRUCT`, `Mat_SizeOfClass()`, `Mat_VarGetSize()`, `matvar_t::nbytes`, and `matvar_t::rank`.

Referenced by `Mat_VarGetSize()`.

1.1.3.19 matvar_t* Mat_VarGetStructField (matvar_t * matvar, void * name_or_index, int opt, int index)

Returns a pointer to the structure field at the given 0-relative index. MAT file version must be 5.

Parameters:

matvar Pointer to the Structure MAT variable

name_or_index Name of the field, or the 1-relative index of the field. If the index is used, it should be the address of an integer variable whose value is the index number.

opt BY_NAME if the name_or_index is the name or BY_INDEX if the index was passed.

index linear index of the structure to find the field of

Returns:

Pointer to the Structure Field on success, NULL on error

References `BY_INDEX`, `BY_NAME`, `matvar_t::data`, `matvar_t::dims`, `matvar_t::name`, `matvar_t::nbytes`, and `matvar_t::rank`.

1.1.3.20 `matvar_t* Mat_VarGetStructs (matvar_t * matvar, int * start, int * stride, int * edge, int copy_fields)`

Finds structures of a structure array given a start, stride, and edge for each dimension. The structures are placed in a new structure array. If `copy_fields` is non-zero, the indexed structures are copied and should be freed, but if `copy_fields` is zero, the indexed structures are pointers to the original, but should still be freed since the `mem_conserve` flag is set so that the structures are not freed. MAT File version must be 5.

Parameters:

matvar Structure matlab variable

start vector of length rank with 0-relative starting coordinates for each dimension.

stride vector of length rank with strides for each dimension.

edge vector of length rank with the number of elements to read in each dimension.

copy_fields 1 to copy the fields, 0 to just set pointers to them. If 0 is used, the fields should not be freed themselves.

Returns:

A new structure with fields indexed from *matvar*.

References `matvar_t::class_type`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::dims`, `MAT_C_STRUCT`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `matvar_t::mem_conserve`, `matvar_t::nbytes`, and `matvar_t::rank`.

1.1.3.21 `matvar_t* Mat_VarGetStructsLinear (matvar_t * matvar, int start, int stride, int edge, int copy_fields)`

Finds structures of a structure array given a single (linear) start, stride, and edge. The structures are placed in a new structure array. If `copy_fields` is non-zero, the indexed structures are copied and should be freed, but if `copy_fields` is zero, the indexed structures are pointers to the original, but should still be freed since the `mem_conserve` flag is set so that the structures are not freed. MAT File version must be 5.

Parameters:

matvar Structure matlab variable

start starting index

stride stride

edge Number of structures to get

copy_fields 1 to copy the fields, 0 to just set pointers to them. If 0 is used, the fields should not be freed themselves.

Returns:

A new structure with fields indexed from *matvar*.

References `matvar_t::data`, `matvar_t::data_size`, `matvar_t::dims`, `Mat_VarDuplicate()`, `matvar_t::mem_conserve`, `matvar_t::nbytes`, and `matvar_t::rank`.

1.1.3.22 `void Mat_VarPrint (matvar_t * matvar, int printdata)`

Prints to stdout the values of the `matvar_t` structure

Parameters:

matvar Pointer to the [matvar_t](#) structure

printdata set to 1 if the Variables data should be printed, else 0

References `matvar_t::fp`, `MAT_FT_MAT4`, `Mat_VarPrint5()`, and `mat_t::version`.

Referenced by `Mat_VarPrint5()`.

1.1.3.23 `matvar_t* Mat_VarRead (mat_t * mat, char * name)`

Reads the next variable in the Matlab MAT file

Parameters:

mat Pointer to the MAT file

name Name of the variable to read

Returns:

Pointer to the [matvar_t](#) structure containing the MAT variable information

References `mat_t::fp`, and `Mat_VarReadInfo()`.

1.1.3.24 `int Mat_VarReadData (mat_t * mat, matvar_t * matvar, void * data, int * start, int * stride, int * edge)`

Reads data from a MAT variable. The variable must have been read by `Mat_VarReadInfo`.

Parameters:

mat MAT file to read data from

matvar MAT variable information

data pointer to store data in (must be pre-allocated)

start array of starting indeces

stride stride of data

edge array specifying the number to read in each direction

Return values:

0 on success

References `MAT_FT_MAT4`, `ReadData5()`, and `mat_t::version`.

1.1.3.25 `int Mat_VarReadDataAll (mat_t * mat, matvar_t * matvar)`

Allocates memory for an reads the data for a given matlab variable.

Parameters:

mat Matlab MAT file structure pointer

matvar Variable whose data is to be read

Returns:

non-zero on error

1.1.3.26 `int Mat_VarReadDataLinear (mat_t * mat, matvar_t * matvar, void * data, int start, int stride, int edge)`

Reads data from a MAT variable using a linear indexing mode. The variable must have been read by `Mat_VarReadInfo`.

Parameters:

mat MAT file to read data from
matvar MAT variable information
data pointer to store data in (must be pre-allocated)
start starting index
stride stride of data
edge number of elements to read

Return values:

0 on success

References `mat_t::byteswap`, `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_NONE`, `COMPRESSION_ZLIB`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `InflateDataType()`, `InflateSkip()`, `InflateSkipData()`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_FT_MAT4`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `matvar_t::rank`, `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadSingleData()`, `mat_t::version`, and `matvar_t::z`.

1.1.3.27 `matvar_t* Mat_VarReadInfo (mat_t * mat, char * name)`

Reads the named variable (or the next variable if `name` is NULL) information (class, flags-complex/global/logical, rank, dimensions, and name) from the Matlab MAT file

Parameters:

mat Pointer to the MAT file
name Name of the variable to read

Returns:

Pointer to the `matvar_t` structure containing the MAT variable information

References `mat_t::bof`, `mat_t::byteswap`, `mat_t::fp`, `Mat_int32Swap()`, `Mat_VarFree()`, `Mat_VarReadNextInfo()`, and `matvar_t::name`.

Referenced by `Mat_VarRead()`.

1.1.3.28 `matvar_t* Mat_VarReadNext (mat_t * mat)`

Reads the next variable in the Matlab MAT file

Parameters:

mat Pointer to the MAT file

Returns:

Pointer to the [matvar_t](#) structure containing the MAT variable information

References `mat_t::fp`, and `Mat_VarReadNextInfo()`.

Referenced by `Mat_VarDelete()`.

1.1.3.29 `matvar_t* Mat_VarReadNextInfo (mat_t * mat)`

Reads the next variable's information (class, flags-complex/global/logical, rank, dimensions, name, etc) from the Matlab MAT file. After reading, the MAT file is positioned past the current variable.

Parameters:

mat Pointer to the MAT file

Returns:

Pointer to the [matvar_t](#) structure containing the MAT variable information

References `Mat_VarReadNextInfo5()`, and `mat_t::version`.

Referenced by `Mat_VarReadInfo()`, `Mat_VarReadNext()`, and `ReadNextFunctionHandle()`.

1.1.3.30 `int Mat_VarWrite (mat_t * mat, matvar_t * matvar, int compress)`

Writes the MAT variable information stored in `matvar` to the given MAT file. The variable will be written to the end of the file.

Parameters:

mat MAT file to write to

matvar MAT variable information to write

compress Whether or not to compress the data (Only valid for version 5 MAT files and variables with numeric data)

Return values:

`0` on success

References `MAT_FT_MAT4`, `mat_t::version`, and `Write5()`.

Referenced by `Mat_VarDelete()`.

1.1.3.31 `int Mat_VarWriteData (mat_t * mat, matvar_t * matvar, void * data, int * start, int * stride, int * edge)`

Writes data to a MAT variable. The variable must have previously been written with `Mat_VarWriteInfo`.

Parameters:

mat MAT file to write to

matvar MAT variable information to write
data pointer to the data to write
start array of starting indeces
stride stride of data
edge array specifying the number to read in each direction

Return values:

0 on success

References `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_NONE`, `COMPRESSION_ZLIB`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `matvar_t::rank`, `WriteCharDataSlab2()`, `WriteData()`, `WriteDataSlab2()`, and `matvar_t::z`.

1.1.3.32 `int Mat_VarWriteInfo (mat_t * mat, matvar_t * matvar)`

Writes the MAT variable information stored in `matvar` to the given MAT file. The variable will be written to the end of the file.

Parameters:

mat MAT file to write to
matvar MAT variable information to write

Return values:

0 on success

References `mat_t::fp`, `MAT_FT_MAT4`, `mat_t::version`, and `WriteInfo5()`.

1.2 Internal Functions

Defines

- `#define swap(a, b) a^=b;b^=a;a^=b`
swap the bytes a and b

Functions

- `int InflateArrayFlags (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the Array Flags Tag and the Array Flags data.
- `int InflateData (mat_t *mat, z_stream *z, void *buf, int nBytes)`
Inflates the data.
- `int InflateDataTag (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the data's tag.
- `int InflateDataType (mat_t *mat, z_stream *z, void *buf)`
Inflates the data's type.
- `int InflateDimensions (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the dimensions tag and the dimensions data.
- `int InflateFieldNameLength (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the structure's fieldname length.
- `int InflateFieldNames (mat_t *mat, matvar_t *matvar, void *buf, int nfields, int fieldname_length, int padding)`
Inflates the structure's fieldnames.
- `int InflateFieldNamesTag (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the structure's fieldname tag.
- `int InflateSkip (mat_t *mat, z_stream *z, int nbytes)`
Inflate the data until nbytes of uncompressed data has been inflated.
- `int InflateSkip2 (mat_t *mat, matvar_t *matvar, int nbytes)`
Inflate the data until nbytes of compressed data has been inflated.
- `int InflateSkipData (mat_t *mat, z_stream *z, int data_type, int len)`
Inflate the data until len elements of compressed data with data type data_type has been inflated.
- `int InflateVarName (mat_t *mat, matvar_t *matvar, void *buf, int N)`
Inflates the variable name.
- `int InflateVarNameTag (mat_t *mat, matvar_t *matvar, void *buf)`
Inflates the variable name tag.

- int [InflateVarTag](#) ([mat_t](#) *mat, [matvar_t](#) *matvar, void *buf)
Inflates the variable's tag.
- double [Mat_doubleSwap](#) (double *a)
swap the bytes of a 4 or 8 byte double-precision float
- float [Mat_floatSwap](#) (float *a)
swap the bytes of a 4 byte single-precision float
- [mat_int16_t](#) [Mat_int16Swap](#) ([mat_int16_t](#) *a)
swap the bytes of a 16-bit signed integer
- [mat_int32_t](#) [Mat_int32Swap](#) ([mat_int32_t](#) *a)
swap the bytes of a 32-bit signed integer
- [mat_int64_t](#) [Mat_int64Swap](#) ([mat_int64_t](#) *a)
swap the bytes of a 64-bit signed integer
- [mat_uint16_t](#) [Mat_uint16Swap](#) ([mat_uint16_t](#) *a)
swap the bytes of a 16-bit unsigned integer
- [mat_uint32_t](#) [Mat_uint32Swap](#) ([mat_uint32_t](#) *a)
swap the bytes of a 32-bit unsigned integer
- [mat_uint64_t](#) [Mat_uint64Swap](#) ([mat_uint64_t](#) *a)
swap the bytes of a 64-bit unsigned integer
- void [Mat_VarPrint5](#) ([matvar_t](#) *matvar, int printdata)
Prints the mat variable.
- [matvar_t](#) * [Mat_VarReadNextInfo5](#) ([mat_t](#) *mat)
Reads the header information for the next MAT variable.
- void [Read5](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Reads the data of a version 5 MAT variable.
- int [ReadCompressedCharData](#) ([mat_t](#) *mat, [z_stream](#) *z, char *data, int data_type, int len)
Reads data of type data_type into a char type.
- int [ReadCompressedDataSlab2](#) ([mat_t](#) *mat, [z_stream](#) *z, void *data, int class_type, int data_type, int *dims, int *start, int *stride, int *edge)
Reads data of type data_type by user-defined dimensions for 2-D data.
- int [ReadCompressedDataSlabN](#) ([mat_t](#) *mat, [z_stream](#) *z, void *data, int class_type, int data_type, int rank, int *dims, int *start, int *stride, int *edge)
Reads data of type data_type by user-defined dimensions.
- int [ReadCompressedDoubleData](#) ([mat_t](#) *mat, [z_stream](#) *z, double *data, int data_type, int len)
Reads data of type data_type into a double type.

- `int ReadCompressedInt16Data (mat_t *mat, z_stream *z, mat_int16_t *data, int data_type, int len)`
Reads data of type data_type into a signed 16-bit integer type.
- `int ReadCompressedInt32Data (mat_t *mat, z_stream *z, mat_int32_t *data, int data_type, int len)`
Reads data of type data_type into a signed 32-bit integer type.
- `int ReadCompressedInt64Data (mat_t *mat, z_stream *z, mat_int64_t *data, int data_type, int len)`
Reads data of type data_type into a signed 64-bit integer type.
- `int ReadCompressedInt8Data (mat_t *mat, z_stream *z, mat_int8_t *data, int data_type, int len)`
Reads data of type data_type into a signed 8-bit integer type.
- `int ReadCompressedSingleData (mat_t *mat, z_stream *z, float *data, int data_type, int len)`
Reads data of type data_type into a float type.
- `int ReadCompressedUInt16Data (mat_t *mat, z_stream *z, mat_uint16_t *data, int data_type, int len)`
Reads data of type data_type into an unsigned 16-bit integer type.
- `int ReadCompressedUInt32Data (mat_t *mat, z_stream *z, mat_uint32_t *data, int data_type, int len)`
Reads data of type data_type into an unsigned 32-bit integer type.
- `int ReadCompressedUInt64Data (mat_t *mat, z_stream *z, mat_uint64_t *data, int data_type, int len)`
Reads data of type data_type into an unsigned 64-bit integer type.
- `int ReadCompressedUInt8Data (mat_t *mat, z_stream *z, mat_uint8_t *data, int data_type, int len)`
Reads data of type data_type into an unsigned 8-bit integer type.
- `int ReadData5 (mat_t *mat, matvar_t *matvar, void *data, int *start, int *stride, int *edge)`
Reads a slab of data from the mat variable matvar.
- `int ReadDataSlab2 (mat_t *mat, void *data, int class_type, int data_type, int *dims, int *start, int *stride, int *edge)`
Reads data of type data_type by user-defined dimensions for 2-D data.
- `int ReadDataSlabN (mat_t *mat, void *data, int class_type, int data_type, int rank, int *dims, int *start, int *stride, int *edge)`
Reads data of type data_type by user-defined dimensions.
- `int ReadDoubleData (mat_t *mat, double *data, int data_type, int len)`
Reads data of type data_type into a double type.
- `int ReadInt16Data (mat_t *mat, mat_int16_t *data, int data_type, int len)`
Reads data of type data_type into a signed 16-bit integer type.
- `int ReadInt32Data (mat_t *mat, mat_int32_t *data, int data_type, int len)`
Reads data of type data_type into a signed 32-bit integer type.

- int [ReadInt64Data](#) ([mat_t](#) *mat, [mat_int64_t](#) *data, int data_type, int len)
Reads data of type data_type into a signed 64-bit integer type.
- int [ReadInt8Data](#) ([mat_t](#) *mat, [mat_int8_t](#) *data, int data_type, int len)
Reads data of type data_type into a signed 8-bit integer type.
- int [ReadNextCell](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Reads the next cell of the cell array in matvar.
- int [ReadNextFunctionHandle](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Reads the function handle data of the function handle in matvar.
- int [ReadNextStructField](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Reads the next struct field of the structure in matvar.
- int [ReadSingleData](#) ([mat_t](#) *mat, float *data, int data_type, int len)
Reads data of type data_type into a float type.
- int [ReadUInt16Data](#) ([mat_t](#) *mat, [mat_uint16_t](#) *data, int data_type, int len)
Reads data of type data_type into an unsigned 16-bit integer type.
- int [ReadUInt32Data](#) ([mat_t](#) *mat, [mat_uint32_t](#) *data, int data_type, int len)
Reads data of type data_type into an unsigned 32-bit integer type.
- int [ReadUInt64Data](#) ([mat_t](#) *mat, [mat_uint64_t](#) *data, int data_type, int len)
Reads data of type data_type into an unsigned 64-bit integer type.
- int [ReadUInt8Data](#) ([mat_t](#) *mat, [mat_uint8_t](#) *data, int data_type, int len)
Reads data of type data_type into an unsigned 8-bit integer type.
- int [Write5](#) ([mat_t](#) *mat, [matvar_t](#) *matvar, int compress)
Writes a matlab variable to a version 5 matlab file.
- int [WriteCellArrayField](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Writes the header and data for an element of a cell array.
- int [WriteCellArrayFieldInfo](#) ([mat_t](#) *mat, [matvar_t](#) *matvar)
Writes the header and blank data for a cell array.
- int [WriteCharData](#) ([mat_t](#) *mat, void *data, int N, int data_type)
Writes data as character data.
- int [WriteCharDataSlab2](#) ([mat_t](#) *mat, void *data, int data_type, int *dims, int *start, int *stride, int *edge)
- size_t [WriteCompressedCellArrayField](#) ([mat_t](#) *mat, [matvar_t](#) *matvar, z_stream *z)
Writes the header and data for a field of a compressed cell array.
- size_t [WriteCompressedCharData](#) ([mat_t](#) *mat, z_stream *z, void *data, int N, int data_type)
Writes data as compressed character data.

- `size_t WriteCompressedStructField (mat_t *mat, matvar_t *matvar, z_stream *z)`
Writes the header and data for a field of a compressed struct array.
- `int WriteDataSlab2 (mat_t *mat, void *data, int data_type, int *dims, int *start, int *stride, int *edge)`
- `int WriteEmptyCharData (mat_t *mat, int N, int data_type)`
Writes empty characters to the MAT file.
- `void WriteInfo5 (mat_t *mat, matvar_t *matvar)`
Writes the variable information and empty data.
- `int WriteStructField (mat_t *mat, matvar_t *matvar)`
Writes the header and data for a field of a struct array.

1.2.1 Detailed Description

1.2.2 Function Documentation

1.2.2.1 `int InflateArrayFlags (mat_t *mat, matvar_t *matvar, void *buf)`

`buf` must hold at least 16 bytes

Parameters:

- mat* Pointer to the MAT file
- matvar* Pointer to the MAT variable
- buf* Pointer to store the 16-byte array flags tag and data

Returns:

Number of bytes read from the file

References `mat_t::fp`, and `matvar_t::z`.

Referenced by `Mat_VarReadNextInfo5()`, `ReadNextCell()`, and `ReadNextStructField()`.

1.2.2.2 `int InflateData (mat_t *mat, z_stream *z, void *buf, int nBytes)`

`buf` must hold at least `nBytes` bytes

Parameters:

- mat* Pointer to the MAT file
- z* zlib compression stream
- buf* Pointer to store the data type
- nBytes* Number of bytes to inflate

Returns:

Number of bytes read from the file

References `mat_t::fp`.

Referenced by `ReadCompressedCharData()`, `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, and `ReadCompressedUInt8Data()`.

1.2.2.3 `int InflateDataTag(mat_t * mat, matvar_t * matvar, void * buf)`

`buf` must hold at least 8 bytes

Parameters:

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the data tag

Returns:

Number of bytes read from the file

References `mat_t::fp`, `matvar_t::name`, and `matvar_t::z`.

1.2.2.4 `int InflateDataType(mat_t * mat, z_stream * z, void * buf)`

`buf` must hold at least 4 bytes

Parameters:

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the data type

Returns:

Number of bytes read from the file

References `mat_t::fp`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, and `ReadData5()`.

1.2.2.5 `int InflateDimensions(mat_t * mat, matvar_t * matvar, void * buf)`

`buf` must hold at least $(8+4*\text{rank})$ bytes where `rank` is the number of dimensions. If the end of the dimensions data is not aligned on an 8-byte boundary, this function eats up those bytes and stores them in `buf`.

Parameters:

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the dimensions flag and data

Returns:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_int32Swap()`, `MAT_T_INT32`, and `matvar_t::z`.

Referenced by `Mat_VarReadNextInfo5()`, `ReadNextCell()`, and `ReadNextStructField()`.

1.2.2.6 int InflateFieldNameLength (mat_t * *mat*, matvar_t * *matvar*, void * *buf*)

buf must hold at least 8 bytes

Parameters:

mat Pointer to the MAT file

matvar Pointer to the MAT variable

buf Pointer to store the fieldname length

Returns:

Number of bytes read from the file

References `mat_t::fp`, and `matvar_t::z`.

Referenced by `ReadNextStructField()`.

1.2.2.7 int InflateFieldNames (mat_t * *mat*, matvar_t * *matvar*, void * *buf*, int *nfields*, int *fieldname_length*, int *padding*)

buf must hold at least `nfields * fieldname_length` bytes

Parameters:

mat Pointer to the MAT file

matvar Pointer to the MAT variable

buf Pointer to store the fieldnames

nfields Number of fields

fieldname_length Maximum length in bytes of each field

padding Number of padding bytes

Returns:

Number of bytes read from the file

References `mat_t::fp`, and `matvar_t::z`.

Referenced by `ReadNextStructField()`.

1.2.2.8 int InflateFieldNamesTag (mat_t * *mat*, matvar_t * *matvar*, void * *buf*)

buf must hold at least 8 bytes

Parameters:

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the fieldname tag

Returns:

Number of bytes read from the file

References `mat_t::fp`, and `matvar_t::z`.

Referenced by `ReadNextStructField()`.

1.2.2.9 int InflateSkip (mat_t * mat, z_stream * z, int nbytes)**Parameters:**

mat Pointer to the MAT file
z zlib compression stream
nbytes Number of uncompressed bytes to skip

Returns:

Number of bytes read from the file

References `mat_t::fp`.

Referenced by `InflateSkipData()`, `Mat_VarReadDataLinear()`, `Read5()`, `ReadData5()`, `ReadNextCell()`, and `ReadNextStructField()`.

1.2.2.10 int InflateSkip2 (mat_t * mat, matvar_t * matvar, int nbytes)**Parameters:**

mat Pointer to the MAT file
z zlib compression stream
nbytes Number of uncompressed bytes to skip

Returns:

Number of bytes read from the file

References `mat_t::fp`, `matvar_t::name`, and `matvar_t::z`.

1.2.2.11 int InflateSkipData (mat_t * mat, z_stream * z, int data_type, int len)**Parameters:**

mat Pointer to the MAT file
z zlib compression stream
data_type Data type (`matio_types` enumerations)
len Number of elements of datatype `data_type` to skip

Returns:

Number of bytes read from the file

References InflateSkip(), MAT_T_DOUBLE, MAT_T_INT16, MAT_T_INT32, MAT_T_INT64, MAT_T_INT8, MAT_T_SINGLE, MAT_T_UINT16, MAT_T_UINT32, MAT_T_UINT64, and MAT_T_UINT8.

Referenced by Mat_VarReadDataLinear(), ReadCompressedDataSlab2(), and ReadCompressedDataSlabN().

1.2.2.12 int InflateVarName (mat_t * *mat*, matvar_t * *matvar*, void * *buf*, int *N*)**Parameters:**

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the variables name
N Number of characters in the name

Returns:

Number of bytes read from the file

References mat_t::fp, and matvar_t::z.

Referenced by Mat_VarReadNextInfo5().

1.2.2.13 int InflateVarNameTag (mat_t * *mat*, matvar_t * *matvar*, void * *buf*)**Parameters:**

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the variables name tag

Returns:

Number of bytes read from the file

References mat_t::fp, and matvar_t::z.

Referenced by Mat_VarReadNextInfo5(), ReadNextCell(), and ReadNextStructField().

1.2.2.14 int InflateVarTag (mat_t * *mat*, matvar_t * *matvar*, void * *buf*)

buf must hold at least 8 bytes

Parameters:

mat Pointer to the MAT file
matvar Pointer to the MAT variable
buf Pointer to store the 8-byte variable tag

Returns:

Number of bytes read from the file

References `mat_t::fp`, and `matvar_t::z`.

Referenced by `Mat_VarReadNextInfo5()`, `ReadNextCell()`, and `ReadNextStructField()`.

1.2.2.15 double Mat_doubleSwap (double * *a*)**Parameters:**

a pointer to integer to swap

Returns:

the swapped integer

References `swap`.

Referenced by `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, `ReadCompressedUInt8Data()`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, and `ReadUInt8Data()`.

1.2.2.16 float Mat_floatSwap (float * *a*)**Parameters:**

a pointer to integer to swap

Returns:

the swapped integer

References `swap`.

Referenced by `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, `ReadCompressedUInt8Data()`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, and `ReadUInt8Data()`.

1.2.2.17 mat_int16_t Mat_int16Swap (mat_int16_t * *a*)**Parameters:**

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by Mat_Open(), ReadCompressedDoubleData(), ReadCompressedInt16Data(), ReadCompressedInt32Data(), ReadCompressedInt64Data(), ReadCompressedInt8Data(), ReadCompressedSingleData(), ReadCompressedUInt16Data(), ReadCompressedUInt32Data(), ReadCompressedUInt64Data(), ReadCompressedUInt8Data(), ReadDoubleData(), ReadInt16Data(), ReadInt32Data(), ReadInt64Data(), ReadInt8Data(), ReadSingleData(), ReadUInt16Data(), ReadUInt32Data(), ReadUInt64Data(), and ReadUInt8Data().

1.2.2.18 mat_int32_t Mat_int32Swap(mat_int32_t * a)

Parameters:

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by InflateDimensions(), Mat_VarReadDataLinear(), Mat_VarReadInfo(), Mat_VarReadNextInfo5(), Read5(), ReadCompressedDoubleData(), ReadCompressedInt16Data(), ReadCompressedInt32Data(), ReadCompressedInt64Data(), ReadCompressedInt8Data(), ReadCompressedSingleData(), ReadCompressedUInt16Data(), ReadCompressedUInt32Data(), ReadCompressedUInt64Data(), ReadCompressedUInt8Data(), ReadData5(), ReadDoubleData(), ReadInt16Data(), ReadInt32Data(), ReadInt64Data(), ReadInt8Data(), ReadSingleData(), ReadUInt16Data(), ReadUInt32Data(), ReadUInt64Data(), ReadUInt8Data(), WriteCellArrayField(), WriteCellArrayFieldInfo(), and WriteStructField().

1.2.2.19 mat_int64_t Mat_int64Swap(mat_int64_t * a)

Parameters:

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by ReadCompressedInt64Data(), ReadCompressedUInt64Data(), ReadInt64Data(), and ReadUInt64Data().

1.2.2.20 mat_uint16_t Mat_uint16Swap(mat_uint16_t * a)

Parameters:

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by ReadCompressedCharData(), ReadCompressedDoubleData(), ReadCompressedInt16Data(), ReadCompressedInt32Data(), ReadCompressedInt64Data(), ReadCompressedInt8Data(), ReadCompressedSingleData(), ReadCompressedUInt16Data(), ReadCompressedUInt32Data(), ReadCompressedUInt64Data(), ReadCompressedUInt8Data(), ReadDoubleData(), ReadInt16Data(), ReadInt32Data(), ReadInt64Data(), ReadInt8Data(), ReadSingleData(), ReadUInt16Data(), ReadUInt32Data(), ReadUInt64Data(), and ReadUInt8Data().

1.2.2.21 `mat_uint32_t Mat_uint32Swap(mat_uint32_t *a)`

Parameters:

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by Mat_VarReadNextInfo5(), Read5(), ReadCompressedDoubleData(), ReadCompressedInt16Data(), ReadCompressedInt32Data(), ReadCompressedInt64Data(), ReadCompressedInt8Data(), ReadCompressedSingleData(), ReadCompressedUInt16Data(), ReadCompressedUInt32Data(), ReadCompressedUInt64Data(), ReadCompressedUInt8Data(), ReadDoubleData(), ReadInt16Data(), ReadInt32Data(), ReadInt64Data(), ReadInt8Data(), ReadNextCell(), ReadNextStructField(), ReadSingleData(), ReadUInt16Data(), ReadUInt32Data(), ReadUInt64Data(), and ReadUInt8Data().

1.2.2.22 `mat_uint64_t Mat_uint64Swap(mat_uint64_t *a)`

Parameters:

a pointer to integer to swap

Returns:

the swapped integer

References swap.

Referenced by ReadCompressedInt64Data(), ReadCompressedUInt64Data(), ReadInt64Data(), and ReadUInt64Data().

1.2.2.23 `void Mat_VarPrint5(matvar_t *matvar, int printdata)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

References matvar_t::class_type, sparse_t::data, matvar_t::data, matvar_t::data_size, matvar_t::data_type, matvar_t::dims, ComplexSplit::Im, sparse_t::ir, matvar_t::isComplex, sparse_t::jc, MAT_C_CELL, MAT_C_CHAR, MAT_C_DOUBLE, MAT_C_INT16, MAT_C_INT32, MAT_C_INT64, MAT_C_INT8,

MAT_C_SINGLE, MAT_C_SPARSE, MAT_C_STRUCT, MAT_C_UINT16, MAT_C_UINT32, MAT_C_UINT64, MAT_C_UINT8, MAT_T_DOUBLE, MAT_T_INT16, MAT_T_INT32, MAT_T_INT64, MAT_T_INT8, MAT_T_SINGLE, MAT_T_UINT16, MAT_T_UINT32, MAT_T_UINT64, MAT_T_UINT8, Mat_VarPrint(), matvar_t::name, matvar_t::nbytes, sparse_t::ndata, sparse_t::njc, matvar_t::rank, and ComplexSplit::Re.

Referenced by Mat_VarPrint().

1.2.2.24 matvar_t* Mat_VarReadNextInfo5 (mat_t * mat)

Parameters:

mat MAT file pointer pointer to the MAT variable or NULL

References mat_t::byteswap, matvar_t::class_type, matvar_t::compression, matvar_t::data, matvar_t::data_size, matvar_t::data_type, matvar_t::datapos, matvar_t::dims, matvar_t::fp, mat_t::fp, matvar_t::fpas, InflateArrayFlags(), InflateDimensions(), InflateVarName(), InflateVarNameTag(), InflateVarTag(), matvar_t::isComplex, matvar_t::isGlobal, matvar_t::isLogical, MAT_C_CELL, MAT_C_FUNCTION, MAT_C_SPARSE, MAT_C_STRUCT, MAT_F_CLASS_T, MAT_F_COMPLEX, MAT_F_GLOBAL, MAT_F_LOGICAL, Mat_int32Swap(), MAT_T_COMPRESSED, MAT_T_INT32, MAT_T_INT8, MAT_T_MATRIX, MAT_T_UINT32, Mat_uint32Swap(), Mat_VarCalloc(), Mat_VarFree(), matvar_t::mem_conserve, matvar_t::name, matvar_t::nbytes, matvar_t::rank, ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), and matvar_t::z.

Referenced by Mat_VarReadNextInfo().

1.2.2.25 void Read5 (mat_t * mat, matvar_t * matvar)

Parameters:

mat MAT file pointer

matvar MAT variable pointer to read the data

References mat_t::byteswap, matvar_t::class_type, matvar_t::compression, COMPRESSION_NONE, COMPRESSION_ZLIB, sparse_t::data, matvar_t::data, matvar_t::data_size, matvar_t::data_type, matvar_t::datapos, matvar_t::dims, matvar_t::fp, mat_t::fp, ComplexSplit::Im, InflateDataType(), InflateSkip(), sparse_t::ir, matvar_t::isComplex, sparse_t::jc, MAT_C_CELL, MAT_C_CHAR, MAT_C_DOUBLE, MAT_C_FUNCTION, MAT_C_INT16, MAT_C_INT32, MAT_C_INT64, MAT_C_INT8, MAT_C_SINGLE, MAT_C_SPARSE, MAT_C_STRUCT, MAT_C_UINT16, MAT_C_UINT32, MAT_C_UINT64, MAT_C_UINT8, Mat_int32Swap(), MAT_T_CELL, MAT_T_DOUBLE, MAT_T_FUNCTION, MAT_T_INT16, MAT_T_INT32, MAT_T_INT64, MAT_T_INT8, MAT_T_SINGLE, MAT_T_STRUCT, MAT_T_UINT16, MAT_T_UINT32, MAT_T_UINT64, MAT_T_UINT8, Mat_uint32Swap(), matvar_t::name, matvar_t::nbytes, sparse_t::ndata, sparse_t::nir, sparse_t::njc, sparse_t::nzmax, matvar_t::rank, ComplexSplit::Re, Read5(), ReadCompressedCharData(), ReadCompressedDoubleData(), ReadCompressedInt16Data(), ReadCompressedInt32Data(), ReadCompressedInt64Data(), ReadCompressedInt8Data(), ReadCompressedSingleData(), ReadCompressedUInt16Data(), ReadCompressedUInt32Data(), ReadCompressedUInt8Data(), ReadDoubleData(), ReadInt16Data(), ReadInt32Data(), ReadInt64Data(), ReadInt8Data(), ReadSingleData(), ReadUInt16Data(), ReadUInt32Data(), ReadUInt8Data(), and matvar_t::z.

Referenced by Read5().

1.2.2.26 `int ReadCompressedCharData (mat_t * mat, z_stream * z, char * data, int data_type, int len)`

Reads from the MAT file *len* compressed elements of data type *data_type* storing them as char's in *data*.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output char values (*len**sizeof(char))

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type *data_type* to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `InflateData()`, `MAT_T_INT16`, `MAT_T_INT8`, `MAT_T_UINT16`, `MAT_T_UINT8`, `MAT_T_UTF8`, and `Mat_uint16Swap()`.

Referenced by `Read5()`, and `ReadCompressedDataSlab2()`.

1.2.2.27 `int ReadCompressedDataSlab2 (mat_t * mat, z_stream * z, void * data, int class_type, int data_type, int * dims, int * start, int * stride, int * edge)`

Parameters:

mat MAT file pointer

z zlib compression stream

data Pointer to store the output data

class_type Type of data class (`matio_classes` enumerations)

data_type Datatype of the stored data (`matio_types` enumerations)

dims Dimensions of the data

start Index to start reading data in each dimension

stride Read every *stride* elements in each dimension

edge Number of elements to read in each dimension

Return values:

Number of bytes read from the file, or -1 on error

References `mat_t::fp`, `InflateSkipData()`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `ReadCompressedCharData()`, `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, and `ReadCompressedUInt8Data()`.

Referenced by `ReadData5()`.

1.2.2.28 `int ReadCompressedDataSlabN (mat_t * mat, z_stream * z, void * data, int class_type, int data_type, int rank, int * dims, int * start, int * stride, int * edge)`

Parameters:

mat MAT file pointer
z zlib compression stream
data Pointer to store the output data
class_type Type of data class (matio_classes enumerations)
data_type Datatype of the stored data (matio_types enumerations)
rank Number of dimensions in the data
dims Dimensions of the data
start Index to start reading data in each dimension
stride Read every *stride* elements in each dimension
edge Number of elements to read in each dimension

Return values:

Number of bytes read from the file, or -1 on error

References `mat_t::fp`, `InflateSkipData()`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, and `ReadCompressedUInt8Data()`.

Referenced by `ReadData5()`.

1.2.2.29 `int ReadCompressedDoubleData (mat_t * mat, z_stream * z, double * data, int data_type, int len)`

Reads from the MAT file *len* compressed elements of data type *data_type* storing them as double's in *data*.

Parameters:

mat MAT file pointer
z Pointer to the zlib stream for inflation
data Pointer to store the output double values (*len**sizeof(double))
data_type one of the `matio_types` enumerations which is the source data type in the file
len Number of elements of type *data_type* to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.30 `int ReadCompressedInt16Data (mat_t * mat, z_stream * z, mat_int16_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as signed 16-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output signed 16-bit integer values (`len*sizeof(mat_int16_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.31 `int ReadCompressedInt32Data (mat_t * mat, z_stream * z, mat_int32_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as signed 32-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output signed 32-bit integer values (`len*sizeof(mat_int32_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.32 `int ReadCompressedInt64Data (mat_t * mat, z_stream * z, mat_int64_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as signed 64-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output signed 64-bit integer values (`len*sizeof(mat_int64_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `Mat_int64Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `Mat_uint16Swap()`, `Mat_uint32Swap()`, and `Mat_uint64Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.33 `int ReadCompressedInt8Data (mat_t * mat, z_stream * z, mat_int8_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as signed 8-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output signed 8-bit integer values (`len*sizeof(mat_int8_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.34 `int ReadCompressedSingleData (mat_t * mat, z_stream * z, float * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as float's in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output float values (`len*sizeof(float)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.35 `int ReadCompressedUInt16Data (mat_t * mat, z_stream * z, mat_uint16_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as unsigned 16-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output `n` unsigned 16-bit integer values (`len*sizeof(mat_uint16_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.36 `int ReadCompressedUInt32Data (mat_t * mat, z_stream * z, mat_uint32_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as unsigned 32-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output unsigned 32-bit integer values (`len*sizeof(mat_uint32_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.37 `int ReadCompressedUInt64Data (mat_t * mat, z_stream * z, mat_uint64_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as unsigned 64-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output unsigned 64-bit integer values (`len*sizeof(mat_uint64_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `Mat_int64Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `Mat_uint16Swap()`, `Mat_uint32Swap()`, and `Mat_uint64Swap()`.

Referenced by `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.38 `int ReadCompressedUInt8Data (mat_t * mat, z_stream * z, mat_uint8_t * data, int data_type, int len)`

Reads from the MAT file `len` compressed elements of data type `data_type` storing them as unsigned 8-bit integers in `data`.

Parameters:

mat MAT file pointer

z Pointer to the zlib stream for inflation

data Pointer to store the output 8-bit integer values (`len*sizeof(mat_uint8_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `InflateData()`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadCompressedDataSlab2()`, and `ReadCompressedDataSlabN()`.

1.2.2.39 `int ReadData5 (mat_t * mat, matvar_t * matvar, void * data, int * start, int * stride, int * edge)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

data pointer to store the read data in (must be of size `edge[0]*...edge[rank-1]*Mat_SizeOfClass(matvar->class_type)`)

start index to start reading data in each dimension

stride write data every `stride` elements in each dimension

edge number of elements to read in each dimension

Return values:

0 on success

References `mat_t::byteswap`, `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_NONE`, `COMPRESSION_ZLIB`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `InflateDataType()`, `InflateSkip()`, `matvar_t::isComplex`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `matvar_t::rank`, `ComplexSplit::Re`, `ReadCompressedDataSlab2()`, `ReadCompressedDataSlabN()`, `ReadDataSlab2()`, `ReadDataSlabN()`, and `matvar_t::z`.

Referenced by `Mat_VarReadData()`.

1.2.2.40 `int ReadDataSlab2 (mat_t * mat, void * data, int class_type, int data_type, int * dims, int * start, int * stride, int * edge)`

Parameters:

mat MAT file pointer
data Pointer to store the output data
class_type Type of data class (matio_classes enumerations)
data_type Datatype of the stored data (matio_types enumerations)
dims Dimensions of the data
start Index to start reading data in each dimension
stride Read every *stride* elements in each dimension
edge Number of elements to read in each dimension

Return values:

Number of bytes read from the file, or -1 on error

References `mat_t::fp`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, and `ReadUInt8Data()`.

Referenced by `ReadData5()`.

1.2.2.41 `int ReadDataSlabN (mat_t * mat, void * data, int class_type, int data_type, int rank, int * dims, int * start, int * stride, int * edge)`

Parameters:

mat MAT file pointer
data Pointer to store the output data
class_type Type of data class (matio_classes enumerations)
data_type Datatype of the stored data (matio_types enumerations)
rank Number of dimensions in the data
dims Dimensions of the data
start Index to start reading data in each dimension
stride Read every *stride* elements in each dimension
edge Number of elements to read in each dimension

Return values:

Number of bytes read from the file, or -1 on error

References `mat_t::fp`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, and `ReadUInt8Data()`.

Referenced by `ReadData5()`.

1.2.2.42 `int ReadDoubleData (mat_t * mat, double * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as double's in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output double values (`len*sizeof(double)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.43 `int ReadInt16Data (mat_t * mat, mat_int16_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as signed 16-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output signed 16-bit integer values (`len*sizeof(mat_int16_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.44 `int ReadInt32Data (mat_t * mat, mat_int32_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as signed 32-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output signed 32-bit integer values (`len*sizeof(mat_int32_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.45 `int ReadInt64Data(mat_t * mat, mat_int64_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as signed 64-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output signed 64-bit integer values (`len*sizeof(mat_int64_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `Mat_int64Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `Mat_uint16Swap()`, `Mat_uint32Swap()`, and `Mat_uint64Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.46 `int ReadInt8Data(mat_t * mat, mat_int8_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as signed 8-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output signed 8-bit integer values (`len*sizeof(mat_int8_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.47 int ReadNextCell (mat_t * *mat*, matvar_t * *matvar*)**Parameters:***mat* MAT file pointer*matvar* MAT variable pointer**Returns:**

Number of bytes read

References `mat_t::byteswap`, `matvar_t::class_type`, `matvar_t::compression`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `matvar_t::fpos`, `InflateArrayFlags()`, `InflateDimensions()`, `InflateSkip()`, `InflateVarNameTag()`, `InflateVarTag()`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `MAT_C_CELL`, `MAT_C_SPARSE`, `MAT_C_STRUCT`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_INT32`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `Mat_uint32Swap()`, `Mat_VarCalloc()`, `Mat_VarFree()`, `matvar_t::name`, `matvar_t::nbytes`, `matvar_t::rank`, `ReadNextCell()`, `ReadNextStructField()`, and `matvar_t::z`.

Referenced by `Mat_VarReadNextInfo5()`, `ReadNextCell()`, and `ReadNextStructField()`.

1.2.2.48 int ReadNextFunctionHandle (mat_t * *mat*, matvar_t * *matvar*)**Parameters:***mat* MAT file pointer*matvar* MAT variable pointer**Returns:**

Number of bytes read

References `matvar_t::data`, `matvar_t::data_size`, `matvar_t::dims`, `Mat_VarReadNextInfo()`, `matvar_t::nbytes`, and `matvar_t::rank`.

Referenced by `Mat_VarReadNextInfo5()`.

1.2.2.49 int ReadNextStructField (mat_t * *mat*, matvar_t * *matvar*)

Reads the next struct fields (fieldname length,names,data headers for all the fields

Parameters:*mat* MAT file pointer*matvar* MAT variable pointer**Returns:**

Number of bytes read

References `mat_t::byteswap`, `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_ZLIB`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `matvar_t::fpos`, `InflateArrayFlags()`, `InflateDimensions()`, `InflateFieldNameLength()`, `InflateFieldNames()`, `InflateFieldNamesTag()`, `InflateSkip()`, `InflateVarNameTag()`, `InflateVarTag()`, `matvar_t::isComplex`, `matvar_t::`

t::isGlobal, matvar_t::isLogical, MAT_C_CELL, MAT_C_SPARSE, MAT_C_STRUCT, MAT_F_CLASS_T, MAT_F_COMPLEX, MAT_F_GLOBAL, MAT_F_LOGICAL, MAT_T_INT32, MAT_T_MATRIX, MAT_T_UINT32, Mat_uint32Swap(), Mat_VarFree(), matvar_t::name, matvar_t::nbytes, matvar_t::rank, ReadNextCell(), ReadNextStructField(), and matvar_t::z.

Referenced by Mat_VarReadNextInfo5(), ReadNextCell(), and ReadNextStructField().

1.2.2.50 int ReadSingleData (mat_t * mat, float * data, int data_type, int len)

Reads from the MAT file `len` elements of data type `data_type` storing them as float's in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output float values (`len*sizeof(float)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Mat_VarReadDataLinear()`, `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.51 int ReadUInt16Data (mat_t * mat, mat_uint16_t * data, int data_type, int len)

Reads from the MAT file `len` elements of data type `data_type` storing them as unsigned 16-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output unsigned 16-bit integer values (`len*sizeof(mat_uint16_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.52 `int ReadUInt32Data(mat_t * mat, mat_uint32_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as unsigned 32-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output unsigned 32-bit integer values (`len*sizeof(mat_uint32_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.53 `int ReadUInt64Data(mat_t * mat, mat_uint64_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as unsigned 64-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output unsigned 64-bit integer values (`len*sizeof(mat_uint64_t)`)

data_type one of the `matio_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `Mat_int64Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT64`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT64`, `MAT_T_UINT8`, `Mat_uint16Swap()`, `Mat_uint32Swap()`, and `Mat_uint64Swap()`.

Referenced by `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.54 `int ReadUInt8Data(mat_t * mat, mat_uint8_t * data, int data_type, int len)`

Reads from the MAT file `len` elements of data type `data_type` storing them as unsigned 8-bit integers in `data`.

Parameters:

mat MAT file pointer

data Pointer to store the output unsigned 8-bit integer values ($\text{len} \times \text{sizeof}(\text{mat_uint8_t})$)

data_type one of the `mat_io_types` enumerations which is the source data type in the file

len Number of elements of type `data_type` to read from the file

Return values:

Number of bytes read from the file

References `mat_t::byteswap`, `mat_t::fp`, `Mat_doubleSwap()`, `Mat_floatSwap()`, `Mat_int16Swap()`, `Mat_int32Swap()`, `MAT_T_DOUBLE`, `MAT_T_INT16`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_SINGLE`, `MAT_T_UINT16`, `MAT_T_UINT32`, `MAT_T_UINT8`, `Mat_uint16Swap()`, and `Mat_uint32Swap()`.

Referenced by `Read5()`, `ReadDataSlab2()`, and `ReadDataSlabN()`.

1.2.2.55 `int Write5(mat_t * mat, matvar_t * matvar, int compress)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

compress option to compress the variable (only works for numeric types)

Return values:

0 on success

References `matvar_t::class_type`, `COMPRESSION_NONE`, `COMPRESSION_ZLIB`, `sparse_t::data`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `sparse_t::ir`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `sparse_t::jc`, `MAT_C_CELL`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_SPARSE`, `MAT_C_STRUCT`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_COMPRESSED`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `matvar_t::name`, `matvar_t::nbytes`, `sparse_t::ndata`, `sparse_t::nir`, `sparse_t::njc`, `matvar_t::rank`, `ComplexSplit::Re`, `WriteCellArrayField()`, `WriteCharData()`, `WriteCompressedCellArrayField()`, `WriteCompressedCharData()`, `WriteCompressedStructField()`, `WriteData()`, `WriteStructField()`, and `matvar_t::z`.

Referenced by `Mat_VarWrite()`.

1.2.2.56 `int WriteCellArrayField(mat_t * mat, matvar_t * matvar)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

Return values:

0 on success

References `mat_t::byteswap`, `matvar_t::class_type`, `sparse_t::data`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `sparse_t::ir`, `matvar_t::isComplex`,

matvar_t::isGlobal, matvar_t::isLogical, sparse_t::jc, MAT_C_CELL, MAT_C_CHAR, MAT_C_DOUBLE, MAT_C_INT16, MAT_C_INT32, MAT_C_INT64, MAT_C_INT8, MAT_C_SINGLE, MAT_C_SPARSE, MAT_C_STRUCT, MAT_C_UINT16, MAT_C_UINT32, MAT_C_UINT64, MAT_C_UINT8, MAT_F_CLASS_T, MAT_F_COMPLEX, MAT_F_GLOBAL, MAT_F_LOGICAL, Mat_int32Swap(), MAT_T_INT32, MAT_T_INT8, MAT_T_MATRIX, MAT_T_UINT32, matvar_t::name, matvar_t::nbytes, sparse_t::ndata, sparse_t::nir, sparse_t::njc, matvar_t::rank, ComplexSplit::Re, WriteCellArrayField(), WriteCharData(), WriteData(), and WriteStructField().

Referenced by Write5(), WriteCellArrayField(), and WriteStructField().

1.2.2.57 int WriteCellArrayFieldInfo (mat_t * *mat*, matvar_t * *matvar*)

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

Returns:

number of bytes written

References mat_t::byteswap, matvar_t::class_type, matvar_t::data, matvar_t::data_size, matvar_t::data_type, matvar_t::datapos, matvar_t::dims, mat_t::fp, matvar_t::isComplex, matvar_t::isGlobal, matvar_t::isLogical, MAT_C_CELL, MAT_C_CHAR, MAT_C_DOUBLE, MAT_C_INT16, MAT_C_INT32, MAT_C_INT64, MAT_C_INT8, MAT_C_SINGLE, MAT_C_UINT16, MAT_C_UINT32, MAT_C_UINT64, MAT_C_UINT8, MAT_F_CLASS_T, MAT_F_COMPLEX, MAT_F_GLOBAL, MAT_F_LOGICAL, Mat_int32Swap(), MAT_T_INT32, MAT_T_INT8, MAT_T_MATRIX, MAT_T_UINT32, matvar_t::name, matvar_t::nbytes, matvar_t::rank, WriteCellArrayFieldInfo(), and WriteEmptyCharData().

Referenced by WriteCellArrayFieldInfo(), and WriteInfo5().

1.2.2.58 int WriteCharData (mat_t * *mat*, void * *data*, int *N*, int *data_type*)

This function uses the knowledge that the data is part of a character class to avoid some pitfalls with Matlab listed below.

- Matlab character data cannot be unsigned 8-bit integers, it needs at least unsigned 16-bit integers

Parameters:

mat MAT file pointer

data character data to write

N Number of elements to write

data_type character data type (enum matio_types)

Returns:

number of bytes written

References mat_t::fp, MAT_T_INT8, MAT_T_UINT16, MAT_T_UINT8, and MAT_T_UTF8.

Referenced by Write5(), WriteCellArrayField(), and WriteStructField().

1.2.2.59 `int WriteCharDataSlab2 (mat_t * mat, void * data, int data_type, int * dims, int * start, int * stride, int * edge)`

Parameters:

Writes a 2-D slab of character data to the MAT file

This function uses the knowledge that the data is part of a character class to avoid some pitfalls with Matlab listed below.

- Matlab character data cannot be unsigned 8-bit integers, it needs at least unsigned 16-bit integers

should return the number of bytes written, but currently returns 0

Parameters:

mat MAT file pointer

data pointer to the slab of data

data_type data type of the data (enum `matio_types`)

dims dimensions of the dataset

start index to start writing the data in each dimension

stride write data every `stride` elements

edge number of elements to write in each dimension

Returns:

number of byteswritten

References `mat_t::fp`, `MAT_T_INT8`, `MAT_T_UINT16`, `MAT_T_UINT8`, and `MAT_T_UTF8`.

Referenced by `Mat_VarWriteData()`.

1.2.2.60 `size_t WriteCompressedCellArrayField (mat_t * mat, matvar_t * matvar, z_stream * z)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

Returns:

number of bytes written to the MAT file

References `matvar_t::class_type`, `sparse_t::data`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapoints`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `sparse_t::ir`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `sparse_t::jc`, `MAT_C_CELL`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_SPARSE`, `MAT_C_STRUCT`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `matvar_t::name`, `matvar_t::nbytes`, `sparse_t::ndata`, `sparse_t::nir`, `sparse_t::njc`, `matvar_t::rank`, `ComplexSplit::Re`, `WriteCompressedCellArrayField()`, `WriteCompressedCharData()`, and `WriteCompressedStructField()`.

Referenced by `Write5()`, `WriteCompressedCellArrayField()`, and `WriteCompressedStructField()`.

1.2.2.61 `size_t WriteCompressedCharData (mat_t * mat, z_stream * z, void * data, int N, int data_type)`

This function uses the knowledge that the data is part of a character class to avoid some pitfalls with Matlab listed below.

- Matlab character data cannot be unsigned 8-bit integers, it needs at least unsigned 16-bit integers

Parameters:

mat MAT file pointer
z pointer to the zlib compression stream
data character data to write
N Number of elements to write
data_type character data type (enum `matio_types`)

Returns:

number of bytes written

References `mat_t::fp`, `MAT_T_INT8`, `MAT_T_UINT16`, `MAT_T_UINT8`, and `MAT_T_UTF8`.

Referenced by `Write5()`, `WriteCompressedCellArrayField()`, and `WriteCompressedStructField()`.

1.2.2.62 `size_t WriteCompressedStructField (mat_t * mat, matvar_t * matvar, z_stream * z)`

Currently does not work for cell arrays or sparse data

Parameters:

mat MAT file pointer
matvar pointer to the mat variable

Returns:

number of bytes written to the MAT file

References `matvar_t::class_type`, `sparse_t::data`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapoints`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `sparse_t::ir`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `sparse_t::jc`, `MAT_C_CELL`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_C_SPARSE`, `MAT_C_STRUCT`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `matvar_t::name`, `matvar_t::nbytes`, `sparse_t::ndata`, `sparse_t::nir`, `sparse_t::njc`, `matvar_t::rank`, `ComplexSplit::Re`, `WriteCompressedCellArrayField()`, `WriteCompressedCharData()`, and `WriteCompressedStructField()`.

Referenced by `Write5()`, `WriteCompressedCellArrayField()`, and `WriteCompressedStructField()`.

1.2.2.63 `int WriteDataSlab2 (mat_t * mat, void * data, int data_type, int * dims, int * start, int * stride, int * edge)`

Parameters:

Writes a 2-D slab of data to the MAT file

should return the number of bytes written, but currently returns 0

Parameters:

mat MAT file pointer
data pointer to the slab of data
data_type data type of the data (enum *matio_types*)
dims dimensions of the dataset
start index to start writing the data in each dimension
stride write data every *stride* elements
edge number of elements to write in each dimension

Returns:

number of byteswritten

References *mat_t::fp*, *MAT_T_DOUBLE*, *MAT_T_INT16*, *MAT_T_INT32*, *MAT_T_INT64*, *MAT_T_INT8*, *MAT_T_SINGLE*, *MAT_T_UINT16*, *MAT_T_UINT32*, *MAT_T_UINT64*, and *MAT_T_UINT8*.

Referenced by *Mat_VarWriteData()*.

1.2.2.64 int WriteEmptyCharData (*mat_t * mat*, int *N*, int *data_type*)

This function uses the knowledge that the data is part of a character class to avoid some pitfalls with Matlab listed below.

- Matlab character data cannot be unsigned 8-bit integers, it needs at least unsigned 16-bit integers

Parameters:

mat MAT file pointer
data character data to write
N Number of elements to write
data_type character data type (enum *matio_types*)

Returns:

number of bytes written

References *mat_t::fp*, *MAT_T_INT8*, *MAT_T_UINT16*, *MAT_T_UINT8*, and *MAT_T_UTF8*.

Referenced by *WriteCellArrayFieldInfo()*, and *WriteInfo5()*.

1.2.2.65 void WriteInfo5 (*mat_t * mat*, *matvar_t * matvar*)

Parameters:

mat MAT file pointer
matvar pointer to the mat variable

References `matvar_t::class_type`, `matvar_t::compression`, `COMPRESSION_NONE`, `COMPRESSION_ZLIB`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::datapos`, `matvar_t::dims`, `mat_t::fp`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `MAT_C_CELL`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_SPARSE`, `MAT_C_STRUCT`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `MAT_T_COMPRESSED`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `matvar_t::name`, `matvar_t::nbytes`, `matvar_t::rank`, `WriteCellArrayFieldInfo()`, `WriteEmptyCharData()`, `WriteInfo5()`, and `matvar_t::z`.

Referenced by `Mat_VarWriteInfo()`, and `WriteInfo5()`.

1.2.2.66 `int WriteStructField(mat_t * mat, matvar_t * matvar)`

Parameters:

mat MAT file pointer

matvar pointer to the mat variable

Return values:

0 on success

References `mat_t::byteswap`, `matvar_t::class_type`, `sparse_t::data`, `matvar_t::data`, `matvar_t::data_size`, `matvar_t::data_type`, `matvar_t::dims`, `mat_t::fp`, `ComplexSplit::Im`, `sparse_t::ir`, `matvar_t::isComplex`, `matvar_t::isGlobal`, `matvar_t::isLogical`, `sparse_t::jc`, `MAT_C_CELL`, `MAT_C_CHAR`, `MAT_C_DOUBLE`, `MAT_C_INT16`, `MAT_C_INT32`, `MAT_C_INT64`, `MAT_C_INT8`, `MAT_C_SINGLE`, `MAT_C_SPARSE`, `MAT_C_STRUCT`, `MAT_C_UINT16`, `MAT_C_UINT32`, `MAT_C_UINT64`, `MAT_C_UINT8`, `MAT_F_CLASS_T`, `MAT_F_COMPLEX`, `MAT_F_GLOBAL`, `MAT_F_LOGICAL`, `Mat_int32Swap()`, `MAT_T_INT32`, `MAT_T_INT8`, `MAT_T_MATRIX`, `MAT_T_UINT32`, `matvar_t::name`, `matvar_t::nbytes`, `sparse_t::ndata`, `sparse_t::nir`, `sparse_t::njc`, `matvar_t::rank`, `ComplexSplit::Re`, `WriteCellArrayField()`, `WriteCharData()`, `WriteData()`, and `WriteStructField()`.

Referenced by `Write5()`, `WriteCellArrayField()`, and `WriteStructField()`.

Chapter 2

Data Structure Documentation

2.1 ComplexSplit Struct Reference

Complex data type using split storage.

Data Fields

- void * [Im](#)
- void * [Re](#)

2.1.1 Detailed Description

Complex data type using split real/imaginary pointers

2.1.2 Field Documentation

2.1.2.1 void* ComplexSplit::Im

Pointer to the imaginary part

Referenced by `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `Mat_VarPrint5()`, `Read5()`, `ReadData5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.1.2.2 void* ComplexSplit::Re

Pointer to the real part

Referenced by `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `Mat_VarPrint5()`, `Read5()`, `ReadData5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.2 `mat_t` Struct Reference

Matlab MAT File information.

Data Fields

- long `bof`
- int `byteswap`
- char * `filename`
- FILE * `fp`
- char * `header`
- int `mode`
- char * `subsys_offset`
- int `version`

2.2.1 Detailed Description

Contains information about a Matlab MAT file

2.2.2 Field Documentation

2.2.2.1 long `mat_t::bof`

Beginning of file not including header

Referenced by `Mat_Create()`, `Mat_Open()`, and `Mat_VarReadInfo()`.

2.2.2.2 int `mat_t::byteswap`

1 if byte swapping is required, 0 else

Referenced by `InflateDimensions()`, `Mat_Create()`, `Mat_Open()`, `Mat_VarReadDataLinear()`, `Mat_VarReadInfo()`, `Mat_VarReadNextInfo5()`, `Read5()`, `ReadCompressedCharData()`, `ReadCompressedDoubleData()`, `ReadCompressedInt16Data()`, `ReadCompressedInt32Data()`, `ReadCompressedInt64Data()`, `ReadCompressedInt8Data()`, `ReadCompressedSingleData()`, `ReadCompressedUInt16Data()`, `ReadCompressedUInt32Data()`, `ReadCompressedUInt64Data()`, `ReadCompressedUInt8Data()`, `ReadData5()`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadNextCell()`, `ReadNextStructField()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, `ReadUInt8Data()`, `WriteCellArrayField()`, `WriteCellArrayFieldInfo()`, and `WriteStructField()`.

2.2.2.3 char* `mat_t::filename`

Name of the file that `fp` points to

Referenced by `Mat_Close()`, `Mat_Create()`, `Mat_Open()`, and `Mat_VarDelete()`.

2.2.2.4 FILE* `mat_t::fp`

Pointer to the MAT file

Referenced by `InflateArrayFlags()`, `InflateData()`, `InflateDataTag()`, `InflateDataType()`, `InflateDimensions()`, `InflateFieldNameLength()`, `InflateFieldNames()`, `InflateFieldNamesTag()`, `InflateSkip()`, `InflateSkip2()`, `InflateVarName()`, `InflateVarNameTag()`, `InflateVarTag()`, `Mat_Close()`, `Mat_Create()`, `Mat_Open()`, `Mat_Rewind()`, `Mat_VarDelete()`, `Mat_VarRead()`, `Mat_VarReadDataLinear()`, `Mat_VarReadInfo()`, `Mat_VarReadNext()`, `Mat_VarReadNextInfo5()`, `Mat_VarWriteData()`, `Mat_VarWriteInfo()`, `Read5()`, `ReadCompressedCharData()`, `ReadCompressedDataSlab2()`, `ReadCompressedDataSlabN()`, `ReadData5()`, `ReadDataSlab2()`, `ReadDataSlabN()`, `ReadDoubleData()`, `ReadInt16Data()`, `ReadInt32Data()`, `ReadInt64Data()`, `ReadInt8Data()`, `ReadNextCell()`, `ReadNextStructField()`, `ReadSingleData()`, `ReadUInt16Data()`, `ReadUInt32Data()`, `ReadUInt64Data()`, `ReadUInt8Data()`, `Write5()`, `WriteCellArrayField()`, `WriteCellArrayFieldInfo()`, `WriteCharData()`, `WriteCharDataSlab2()`, `WriteCompressedCellArrayField()`, `WriteCompressedCharData()`, `WriteCompressedStructField()`, `WriteData()`, `WriteDataSlab2()`, `WriteEmptyCharData()`, `WriteInfo5()`, and `WriteStructField()`.

2.2.2.5 char* mat_t::header

MAT File header string

Referenced by `Mat_Close()`, `Mat_Create()`, `Mat_Open()`, and `Mat_VarDelete()`.

2.2.2.6 int mat_t::mode

Access mode

Referenced by `Mat_Create()`, `Mat_Open()`, and `Mat_VarDelete()`.

2.2.2.7 char* mat_t::subsys_offset

offset

Referenced by `Mat_Close()`, `Mat_Create()`, and `Mat_Open()`.

2.2.2.8 int mat_t::version

MAT File version

Referenced by `Mat_Create()`, `Mat_Open()`, `Mat_Rewind()`, `Mat_VarPrint()`, `Mat_VarReadData()`, `Mat_VarReadDataLinear()`, `Mat_VarReadNextInfo()`, `Mat_VarWrite()`, and `Mat_VarWriteInfo()`.

2.3 matvar_t Struct Reference

Matlab variable information.

Data Fields

- int [class_type](#)
- int [compression](#)
- void * [data](#)
- int [data_size](#)
- int [data_type](#)
- long [datapos](#)
- int * [dims](#)
- [mat_t](#) * [fp](#)
- long [fpos](#)
- int [isComplex](#)
- int [isGlobal](#)
- int [isLogical](#)
- int [mem_conserve](#)
- char * [name](#)
- int [nbytes](#)
- int [rank](#)
- z_stream * [z](#)

2.3.1 Detailed Description

Contains information about a Matlab variable

2.3.2 Field Documentation

2.3.2.1 int matvar_t::class_type

Class type in Matlab(mxDOUBLE_CLASS, etc)

Referenced by [Mat_VarCalloc\(\)](#), [Mat_VarCreate\(\)](#), [Mat_VarDuplicate\(\)](#), [Mat_VarFree\(\)](#), [Mat_VarGetNumberOfFields\(\)](#), [Mat_VarGetSize\(\)](#), [Mat_VarGetStructs\(\)](#), [Mat_VarPrint5\(\)](#), [Mat_VarReadDataLinear\(\)](#), [Mat_VarReadNextInfo5\(\)](#), [Mat_VarWriteData\(\)](#), [Read5\(\)](#), [ReadData5\(\)](#), [ReadNextCell\(\)](#), [ReadNextStructField\(\)](#), [Write5\(\)](#), [WriteCellArrayField\(\)](#), [WriteCellArrayFieldInfo\(\)](#), [WriteCompressedCellArrayField\(\)](#), [WriteCompressedStructField\(\)](#), [WriteInfo5\(\)](#), and [WriteStructField\(\)](#).

2.3.2.2 int matvar_t::compression

Compression (0=>None,1=>ZLIB)

Referenced by [Mat_VarCalloc\(\)](#), [Mat_VarCreate\(\)](#), [Mat_VarDuplicate\(\)](#), [Mat_VarFree\(\)](#), [Mat_VarReadDataLinear\(\)](#), [Mat_VarReadNextInfo5\(\)](#), [Mat_VarWriteData\(\)](#), [Read5\(\)](#), [ReadData5\(\)](#), [ReadNextCell\(\)](#), [ReadNextStructField\(\)](#), and [WriteInfo5\(\)](#).

2.3.2.3 void* matvar_t::data

Pointer to the data

Referenced by Mat_VarAddStructField(), Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarFree(), Mat_VarGetCell(), Mat_VarGetCells(), Mat_VarGetCellsLinear(), Mat_VarGetSize(), Mat_VarGetStructField(), Mat_VarGetStructs(), Mat_VarGetStructsLinear(), Mat_VarPrint5(), Mat_VarReadNextInfo5(), Read5(), ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.4 int matvar_t::data_size

Bytes / element for the data

Referenced by Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarFree(), Mat_VarGetNumberOfFields(), Mat_VarGetSize(), Mat_VarGetStructs(), Mat_VarGetStructsLinear(), Mat_VarPrint5(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Read5(), ReadData5(), ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.5 int matvar_t::data_type

Data type(MAT_T_*)

Referenced by Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarPrint5(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Mat_VarWriteData(), Read5(), ReadData5(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.6 long matvar_t::datapos

Offset from the beginning of the MAT file to the data

Referenced by Mat_VarCalloc(), Mat_VarDuplicate(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Mat_VarWriteData(), Read5(), ReadData5(), ReadNextCell(), ReadNextStructField(), Write5(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), and WriteInfo5().

2.3.2.7 int* matvar_t::dims

Array of lengths for each dimension

Referenced by Mat_VarAddStructField(), Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarFree(), Mat_VarGetCell(), Mat_VarGetCells(), Mat_VarGetNumberOfFields(), Mat_VarGetSize(), Mat_VarGetStructField(), Mat_VarGetStructs(), Mat_VarGetStructsLinear(), Mat_VarPrint5(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Mat_VarWriteData(), Read5(), ReadData5(), ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.8 `mat_t* matvar_t::fp`

Pointer to the MAT file structure ([mat_t](#))

Referenced by `Mat_VarCalloc()`, `Mat_VarPrint()`, `Mat_VarReadNextInfo5()`, and `Read5()`.

2.3.2.9 `long matvar_t::fpos`

Offset from the beginning of the MAT file to the variable

Referenced by `Mat_VarCalloc()`, `Mat_VarDuplicate()`, `Mat_VarReadNextInfo5()`, `ReadNextCell()`, and `ReadNextStructField()`.

2.3.2.10 `int matvar_t::isComplex`

non-zero if the data is complex, 0 if real

Referenced by `Mat_VarCalloc()`, `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `Mat_VarPrint5()`, `Mat_VarReadNextInfo5()`, `Read5()`, `ReadData5()`, `ReadNextCell()`, `ReadNextStructField()`, `Write5()`, `WriteCellArrayField()`, `WriteCellArrayFieldInfo()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, `WriteInfo5()`, and `WriteStructField()`.

2.3.2.11 `int matvar_t::isGlobal`

non-zero if the variable is global

Referenced by `Mat_VarCalloc()`, `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarReadNextInfo5()`, `ReadNextCell()`, `ReadNextStructField()`, `Write5()`, `WriteCellArrayField()`, `WriteCellArrayFieldInfo()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, `WriteInfo5()`, and `WriteStructField()`.

2.3.2.12 `int matvar_t::isLogical`

non-zero if the variable is logical

Referenced by `Mat_VarCalloc()`, `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarReadNextInfo5()`, `ReadNextCell()`, `ReadNextStructField()`, `Write5()`, `WriteCellArrayField()`, `WriteCellArrayFieldInfo()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, `WriteInfo5()`, and `WriteStructField()`.

2.3.2.13 `int matvar_t::mem_conserve`

1 if Memory was conserved with data

Referenced by `Mat_VarCalloc()`, `Mat_VarCreate()`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `Mat_VarGetStructs()`, `Mat_VarGetStructsLinear()`, and `Mat_VarReadNextInfo5()`.

2.3.2.14 `char* matvar_t::name`

Name of the variable

Referenced by `InflateDataTag()`, `InflateSkip2()`, `Mat_VarCalloc()`, `Mat_VarCreate()`, `Mat_VarDelete()`, `Mat_VarDuplicate()`, `Mat_VarFree()`, `Mat_VarGetStructField()`, `Mat_VarPrint5()`, `Mat_VarReadInfo()`,

Mat_VarReadNextInfo5(), Read5(), ReadNextCell(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.15 int matvar_t::nbytes

Number of bytes for the MAT variable

Referenced by Mat_VarAddStructField(), Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarFree(), Mat_VarGetNumberOfFields(), Mat_VarGetSize(), Mat_VarGetStructField(), Mat_VarGetStructs(), Mat_VarGetStructsLinear(), Mat_VarPrint5(), Mat_VarReadNextInfo5(), Read5(), ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.16 int matvar_t::rank

Rank (Number of dimensions) of the data

Referenced by Mat_VarAddStructField(), Mat_VarCalloc(), Mat_VarCreate(), Mat_VarDuplicate(), Mat_VarGetCell(), Mat_VarGetCells(), Mat_VarGetCellsLinear(), Mat_VarGetNumberOfFields(), Mat_VarGetSize(), Mat_VarGetStructField(), Mat_VarGetStructs(), Mat_VarGetStructsLinear(), Mat_VarPrint5(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Mat_VarWriteData(), Read5(), ReadData5(), ReadNextCell(), ReadNextFunctionHandle(), ReadNextStructField(), Write5(), WriteCellArrayField(), WriteCellArrayFieldInfo(), WriteCompressedCellArrayField(), WriteCompressedStructField(), WriteInfo5(), and WriteStructField().

2.3.2.17 z_stream* matvar_t::z

zlib compression state

Referenced by InflateArrayFlags(), InflateDataTag(), InflateDimensions(), InflateFieldNameLength(), InflateFieldNames(), InflateFieldNamesTag(), InflateSkip2(), InflateVarName(), InflateVarNameTag(), InflateVarTag(), Mat_VarCalloc(), Mat_VarDuplicate(), Mat_VarFree(), Mat_VarReadDataLinear(), Mat_VarReadNextInfo5(), Mat_VarWriteData(), Read5(), ReadData5(), ReadNextCell(), ReadNextStructField(), Write5(), and WriteInfo5().

2.4 sparse_t Struct Reference

sparse data information

Data Fields

- void * [data](#)
- int * [ir](#)
- int * [jc](#)
- int [ndata](#)
- int [nir](#)
- int [njc](#)
- int [nzmax](#)

2.4.1 Detailed Description

Contains information and data for a sparse matrix

2.4.2 Field Documentation

2.4.2.1 void* sparse_t::data

Array of data elements

Referenced by `Mat_VarFree()`, `Mat_VarPrint5()`, `Read5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.4.2.2 int* sparse_t::ir

Array of size `nzmax` where `ir[k]` is the row of `data[k]`. $0 \leq k \leq nzmax$

Referenced by `Mat_VarFree()`, `Mat_VarPrint5()`, `Read5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.4.2.3 int* sparse_t::jc

Array size `N+1` (`N` is number of columns) with `jc[k]` being the index into `ir/data` of the first non-zero element for row `k`.

Referenced by `Mat_VarFree()`, `Mat_VarPrint5()`, `Read5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.4.2.4 int sparse_t::ndata

Number of complex/real data values

Referenced by `Mat_VarPrint5()`, `Read5()`, `Write5()`, `WriteCellArrayField()`, `WriteCompressedCellArrayField()`, `WriteCompressedStructField()`, and `WriteStructField()`.

2.4.2.5 int sparse_t::nir

number of elements in ir

Referenced by Read5(), Write5(), WriteCellArrayField(), WriteCompressedCellArrayField(), WriteCompressedStructField(), and WriteStructField().

2.4.2.6 int sparse_t::njc

Number of elements in jc

Referenced by Mat_VarPrint5(), Read5(), Write5(), WriteCellArrayField(), WriteCompressedCellArrayField(), WriteCompressedStructField(), and WriteStructField().

2.4.2.7 int sparse_t::nzmax

Maximum number of non-zero elements

Referenced by Read5().

Index

- bof
 - mat_t, [54](#)
- BY_INDEX
 - MAT, [6](#)
- BY_NAME
 - MAT, [6](#)
- byteswap
 - mat_t, [54](#)
- class_type
 - matvar_t, [56](#)
- ComplexSplit, [53](#)
 - Im, [53](#)
 - Re, [53](#)
- compression
 - matvar_t, [56](#)
- COMPRESSION_NONE
 - MAT, [7](#)
- COMPRESSION_ZLIB
 - MAT, [7](#)
- data
 - matvar_t, [56](#)
 - sparse_t, [60](#)
- data_size
 - matvar_t, [57](#)
- data_type
 - matvar_t, [57](#)
- datapos
 - matvar_t, [57](#)
- dims
 - matvar_t, [57](#)
- filename
 - mat_t, [54](#)
- fp
 - mat_t, [54](#)
 - matvar_t, [57](#)
- fpos
 - matvar_t, [58](#)
- header
 - mat_t, [55](#)
- Im
 - ComplexSplit, [53](#)
- InflateArrayFlags
 - mat_internal, [25](#)
- InflateData
 - mat_internal, [25](#)
- InflateDataTag
 - mat_internal, [26](#)
- InflateDataType
 - mat_internal, [26](#)
- InflateDimensions
 - mat_internal, [26](#)
- InflateFieldNameLength
 - mat_internal, [27](#)
- InflateFieldNames
 - mat_internal, [27](#)
- InflateFieldNamesTag
 - mat_internal, [27](#)
- InflateSkip
 - mat_internal, [28](#)
- InflateSkip2
 - mat_internal, [28](#)
- InflateSkipData
 - mat_internal, [28](#)
- InflateVarName
 - mat_internal, [29](#)
- InflateVarNameTag
 - mat_internal, [29](#)
- InflateVarTag
 - mat_internal, [29](#)
- Internal Functions, [21](#)
- ir
 - sparse_t, [60](#)
- isComplex
 - matvar_t, [58](#)
- isGlobal
 - matvar_t, [58](#)
- isLogical
 - matvar_t, [58](#)
- jc
 - sparse_t, [60](#)
- MAT
 - BY_INDEX, [6](#)
 - BY_NAME, [6](#)
 - COMPRESSION_NONE, [7](#)

COMPRESSION_ZLIB, 7
 MAT_ACC_RDONLY, 6
 MAT_ACC_RDWR, 6
 MAT_C_CELL, 7
 MAT_C_CHAR, 7
 MAT_C_DOUBLE, 7
 MAT_C_FUNCTION, 7
 MAT_C_INT16, 7
 MAT_C_INT32, 7
 MAT_C_INT64, 7
 MAT_C_INT8, 7
 MAT_C_OBJECT, 7
 MAT_C_SINGLE, 7
 MAT_C_SPARSE, 7
 MAT_C_STRUCT, 7
 MAT_C_UINT16, 7
 MAT_C_UINT32, 7
 MAT_C_UINT64, 7
 MAT_C_UINT8, 7
 MAT_F_CLASS_T, 8
 MAT_F_COMPLEX, 8
 MAT_F_GLOBAL, 8
 MAT_F_LOGICAL, 8
 MAT_FT_MAT4, 7
 MAT_FT_MAT5, 7
 MAT_T_ARRAY, 8
 MAT_T_CELL, 8
 MAT_T_COMPRESSED, 8
 MAT_T_DOUBLE, 8
 MAT_T_FUNCTION, 8
 MAT_T_INT16, 8
 MAT_T_INT32, 8
 MAT_T_INT64, 8
 MAT_T_INT8, 8
 MAT_T_MATRIX, 8
 MAT_T_SINGLE, 8
 MAT_T_STRING, 8
 MAT_T_STRUCT, 8
 MAT_T_UINT16, 8
 MAT_T_UINT32, 8
 MAT_T_UINT64, 8
 MAT_T_UINT8, 8
 MAT_T_UNKNOWN, 8
 MAT_T_UTF16, 8
 MAT_T_UTF32, 8
 MAT_T_UTF8, 8
 mat_acc, 6
 Mat_CalcSingleSubscript, 8
 Mat_CalcSubscripts, 9
 Mat_Close, 9
 Mat_Create, 9
 mat_ft, 6
 Mat_Open, 10
 Mat_Rewind, 10
 Mat_SizeOfClass, 10
 Mat_VarAddStructField, 11
 Mat_VarCalloc, 11
 Mat_VarCreate, 11
 Mat_VarDelete, 12
 Mat_VarDuplicate, 13
 Mat_VarFree, 13
 Mat_VarGetCell, 13
 Mat_VarGetCells, 14
 Mat_VarGetCellsLinear, 14
 Mat_VarGetNumberOfFields, 14
 Mat_VarGetSize, 15
 Mat_VarGetStructField, 15
 Mat_VarGetStructs, 15
 Mat_VarGetStructsLinear, 16
 Mat_VarPrint, 16
 Mat_VarRead, 17
 Mat_VarReadData, 17
 Mat_VarReadDataAll, 17
 Mat_VarReadDataLinear, 17
 Mat_VarReadInfo, 18
 Mat_VarReadNext, 18
 Mat_VarReadNextInfo, 19
 Mat_VarWrite, 19
 Mat_VarWriteData, 19
 Mat_VarWriteInfo, 20
 matio_classes, 7
 matio_compression, 7
 matio_flags, 7
 matio_types, 8
 MAT_ACC_RDONLY
 MAT, 6
 MAT_ACC_RDWR
 MAT, 6
 MAT_C_CELL
 MAT, 7
 MAT_C_CHAR
 MAT, 7
 MAT_C_DOUBLE
 MAT, 7
 MAT_C_FUNCTION
 MAT, 7
 MAT_C_INT16
 MAT, 7
 MAT_C_INT32
 MAT, 7
 MAT_C_INT64
 MAT, 7
 MAT_C_INT8
 MAT, 7
 MAT_C_OBJECT
 MAT, 7
 MAT_C_SINGLE
 MAT, 7

- MAT_C_SPARSE
 - MAT, [7](#)
- MAT_C_STRUCT
 - MAT, [7](#)
- MAT_C_UINT16
 - MAT, [7](#)
- MAT_C_UINT32
 - MAT, [7](#)
- MAT_C_UINT64
 - MAT, [7](#)
- MAT_C_UINT8
 - MAT, [7](#)
- MAT_F_CLASS_T
 - MAT, [8](#)
- MAT_F_COMPLEX
 - MAT, [8](#)
- MAT_F_GLOBAL
 - MAT, [8](#)
- MAT_F_LOGICAL
 - MAT, [8](#)
- MAT_FT_MAT4
 - MAT, [7](#)
- MAT_FT_MAT5
 - MAT, [7](#)
- MAT_T_ARRAY
 - MAT, [8](#)
- MAT_T_CELL
 - MAT, [8](#)
- MAT_T_COMPRESSED
 - MAT, [8](#)
- MAT_T_DOUBLE
 - MAT, [8](#)
- MAT_T_FUNCTION
 - MAT, [8](#)
- MAT_T_INT16
 - MAT, [8](#)
- MAT_T_INT32
 - MAT, [8](#)
- MAT_T_INT64
 - MAT, [8](#)
- MAT_T_INT8
 - MAT, [8](#)
- MAT_T_MATRIX
 - MAT, [8](#)
- MAT_T_SINGLE
 - MAT, [8](#)
- MAT_T_STRING
 - MAT, [8](#)
- MAT_T_STRUCT
 - MAT, [8](#)
- MAT_T_UINT16
 - MAT, [8](#)
- MAT_T_UINT32
 - MAT, [8](#)
- MAT_T_UINT64
 - MAT, [8](#)
- MAT_T_UINT8
 - MAT, [8](#)
- MAT_T_UNKNOWN
 - MAT, [8](#)
- MAT_T_UTF16
 - MAT, [8](#)
- MAT_T_UTF32
 - MAT, [8](#)
- MAT_T_UTF8
 - MAT, [8](#)
- mat_acc
 - MAT, [6](#)
- Mat_CalcSingleSubscript
 - MAT, [8](#)
- Mat_CalcSubscripts
 - MAT, [9](#)
- Mat_Close
 - MAT, [9](#)
- Mat_Create
 - MAT, [9](#)
- Mat_doubleSwap
 - mat_internal, [30](#)
- Mat_floatSwap
 - mat_internal, [30](#)
- mat_ft
 - MAT, [6](#)
- Mat_int16Swap
 - mat_internal, [30](#)
- Mat_int32Swap
 - mat_internal, [31](#)
- Mat_int64Swap
 - mat_internal, [31](#)
- mat_internal
 - InflateArrayFlags, [25](#)
 - InflateData, [25](#)
 - InflateDataTag, [26](#)
 - InflateDataType, [26](#)
 - InflateDimensions, [26](#)
 - InflateFieldNameLength, [27](#)
 - InflateFieldNames, [27](#)
 - InflateFieldNamesTag, [27](#)
 - InflateSkip, [28](#)
 - InflateSkip2, [28](#)
 - InflateSkipData, [28](#)
 - InflateVarName, [29](#)
 - InflateVarNameTag, [29](#)
 - InflateVarTag, [29](#)
 - Mat_doubleSwap, [30](#)
 - Mat_floatSwap, [30](#)
 - Mat_int16Swap, [30](#)
 - Mat_int32Swap, [31](#)
 - Mat_int64Swap, [31](#)

- Mat_uint16Swap, 31
- Mat_uint32Swap, 32
- Mat_uint64Swap, 32
- Mat_VarPrint5, 32
- Mat_VarReadNextInfo5, 33
- Read5, 33
- ReadCompressedCharData, 33
- ReadCompressedDataSlab2, 34
- ReadCompressedDataSlabN, 34
- ReadCompressedDoubleData, 35
- ReadCompressedInt16Data, 35
- ReadCompressedInt32Data, 36
- ReadCompressedInt64Data, 36
- ReadCompressedInt8Data, 37
- ReadCompressedSingleData, 37
- ReadCompressedUInt16Data, 38
- ReadCompressedUInt32Data, 38
- ReadCompressedUInt64Data, 39
- ReadCompressedUInt8Data, 39
- ReadData5, 40
- ReadDataSlab2, 40
- ReadDataSlabN, 41
- ReadDoubleData, 41
- ReadInt16Data, 42
- ReadInt32Data, 42
- ReadInt64Data, 43
- ReadInt8Data, 43
- ReadNextCell, 43
- ReadNextFunctionHandle, 44
- ReadNextStructField, 44
- ReadSingleData, 45
- ReadUInt16Data, 45
- ReadUInt32Data, 45
- ReadUInt64Data, 46
- ReadUInt8Data, 46
- Write5, 47
- WriteCellArrayField, 47
- WriteCellArrayFieldInfo, 48
- WriteCharData, 48
- WriteCharDataSlab2, 48
- WriteCompressedCellArrayField, 49
- WriteCompressedCharData, 49
- WriteCompressedStructField, 50
- WriteDataSlab2, 50
- WriteEmptyCharData, 51
- WriteInfo5, 51
- WriteStructField, 52
- Mat_Open
 - MAT, 10
- Mat_Rewind
 - MAT, 10
- Mat_SizeOfClass
 - MAT, 10
- mat_t, 54
 - bof, 54
 - byteswap, 54
 - filename, 54
 - fp, 54
 - header, 55
 - mode, 55
 - subsys_offset, 55
 - version, 55
- Mat_uint16Swap
 - mat_internal, 31
- Mat_uint32Swap
 - mat_internal, 32
- Mat_uint64Swap
 - mat_internal, 32
- Mat_VarAddStructField
 - MAT, 11
- Mat_VarCalloc
 - MAT, 11
- Mat_VarCreate
 - MAT, 11
- Mat_VarDelete
 - MAT, 12
- Mat_VarDuplicate
 - MAT, 13
- Mat_VarFree
 - MAT, 13
- Mat_VarGetCell
 - MAT, 13
- Mat_VarGetCells
 - MAT, 14
- Mat_VarGetCellsLinear
 - MAT, 14
- Mat_VarGetNumberOfFields
 - MAT, 14
- Mat_VarGetSize
 - MAT, 15
- Mat_VarGetStructField
 - MAT, 15
- Mat_VarGetStructs
 - MAT, 15
- Mat_VarGetStructsLinear
 - MAT, 16
- Mat_VarPrint
 - MAT, 16
- Mat_VarPrint5
 - mat_internal, 32
- Mat_VarRead
 - MAT, 17
- Mat_VarReadData
 - MAT, 17
- Mat_VarReadDataAll
 - MAT, 17
- Mat_VarReadDataLinear
 - MAT, 17

- Mat_VarReadInfo
 - MAT, 18
- Mat_VarReadNext
 - MAT, 18
- Mat_VarReadNextInfo
 - MAT, 19
- Mat_VarReadNextInfo5
 - mat_internal, 33
- Mat_VarWrite
 - MAT, 19
- Mat_VarWriteData
 - MAT, 19
- Mat_VarWriteInfo
 - MAT, 20
- matio_classes
 - MAT, 7
- matio_compression
 - MAT, 7
- matio_flags
 - MAT, 7
- matio_types
 - MAT, 8
- Matlab MAT File I/O Library, 3
- matvar_t, 56
 - class_type, 56
 - compression, 56
 - data, 56
 - data_size, 57
 - data_type, 57
 - datapos, 57
 - dims, 57
 - fp, 57
 - fpos, 58
 - isComplex, 58
 - isGlobal, 58
 - isLogical, 58
 - mem_conserve, 58
 - name, 58
 - nbytes, 59
 - rank, 59
 - z, 59
- mem_conserve
 - matvar_t, 58
- mode
 - mat_t, 55
- name
 - matvar_t, 58
- nbytes
 - matvar_t, 59
- ndata
 - sparse_t, 60
- nir
 - sparse_t, 60
- njc
 - sparse_t, 61
- nzmax
 - sparse_t, 61
- rank
 - matvar_t, 59
- Re
 - ComplexSplit, 53
- Read5
 - mat_internal, 33
- ReadCompressedCharData
 - mat_internal, 33
- ReadCompressedDataSlab2
 - mat_internal, 34
- ReadCompressedDataSlabN
 - mat_internal, 34
- ReadCompressedDoubleData
 - mat_internal, 35
- ReadCompressedInt16Data
 - mat_internal, 35
- ReadCompressedInt32Data
 - mat_internal, 36
- ReadCompressedInt64Data
 - mat_internal, 36
- ReadCompressedInt8Data
 - mat_internal, 37
- ReadCompressedSingleData
 - mat_internal, 37
- ReadCompressedUInt16Data
 - mat_internal, 38
- ReadCompressedUInt32Data
 - mat_internal, 38
- ReadCompressedUInt64Data
 - mat_internal, 39
- ReadCompressedUInt8Data
 - mat_internal, 39
- ReadData5
 - mat_internal, 40
- ReadDataSlab2
 - mat_internal, 40
- ReadDataSlabN
 - mat_internal, 41
- ReadDoubleData
 - mat_internal, 41
- ReadInt16Data
 - mat_internal, 42
- ReadInt32Data
 - mat_internal, 42
- ReadInt64Data
 - mat_internal, 43
- ReadInt8Data
 - mat_internal, 43
- ReadNextCell

- mat_internal, [43](#)
- ReadNextFunctionHandle
 - mat_internal, [44](#)
- ReadNextStructField
 - mat_internal, [44](#)
- ReadSingleData
 - mat_internal, [45](#)
- ReadUInt16Data
 - mat_internal, [45](#)
- ReadUInt32Data
 - mat_internal, [45](#)
- ReadUInt64Data
 - mat_internal, [46](#)
- ReadUInt8Data
 - mat_internal, [46](#)
- sparse_t, [60](#)
 - data, [60](#)
 - ir, [60](#)
 - jc, [60](#)
 - ndata, [60](#)
 - nir, [60](#)
 - njc, [61](#)
 - nzmax, [61](#)
- subsys_offset
 - mat_t, [55](#)
- version
 - mat_t, [55](#)
- Write5
 - mat_internal, [47](#)
- WriteCellArrayField
 - mat_internal, [47](#)
- WriteCellArrayFieldInfo
 - mat_internal, [48](#)
- WriteCharData
 - mat_internal, [48](#)
- WriteCharDataSlab2
 - mat_internal, [48](#)
- WriteCompressedCellArrayField
 - mat_internal, [49](#)
- WriteCompressedCharData
 - mat_internal, [49](#)
- WriteCompressedStructField
 - mat_internal, [50](#)
- WriteDataSlab2
 - mat_internal, [50](#)
- WriteEmptyCharData
 - mat_internal, [51](#)
- WriteInfo5
 - mat_internal, [51](#)
- WriteStructField
 - mat_internal, [52](#)
- z
 - matvar_t, [59](#)