

# Design Space Toolbox

## 2.0.1

Generated by Doxygen 1.6.3

Wed Feb 22 14:44:50 2012



# Contents

<b>1</b>	<b>Todo List</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Macros to manipulate dictionary nodes. . . . .	9
5.1.1	Detailed Description . . . . .	9
5.1.2	Define Documentation . . . . .	9
5.1.2.1	dsInternalDictionarySetValue . . . . .	9
5.1.2.2	dsInternalDictionaryValue . . . . .	9
5.2	Messages for DS Errors. . . . .	10
5.2.1	Detailed Description . . . . .	11
5.3	Actions for DS Errors. . . . .	12
5.3.1	Detailed Description . . . . .	12
5.4	DSGMAACCESSORS . . . . .	13
5.4.1	Detailed Description . . . . .	13
5.5	DS_IO_TAG_TYPES . . . . .	14
5.6	Options for JSON conversion of DSCase object. . . . .	15
5.6.1	Detailed Description . . . . .	15
5.7	Options for JSON conversion of DSSSystem object. . . . .	16
5.7.1	Detailed Description . . . . .	16
5.8	DSSSysACCESSORS . . . . .	17
5.8.1	Detailed Description . . . . .	17

5.9	Macros to manipulate variables. . . . .	18
5.9.1	Detailed Description . . . . .	18
5.9.2	Define Documentation . . . . .	18
5.9.2.1	DSVariableName . . . . .	18
5.9.2.2	DSVariableSetValue . . . . .	18
5.9.2.3	DSVariableValue . . . . .	18
<b>6</b>	<b>Data Structure Documentation</b>	<b>19</b>
6.1	_varDictionary Struct Reference . . . . .	19
6.1.1	Detailed Description . . . . .	19
6.2	base_info Union Reference . . . . .	21
6.3	ds_parallelstack_t Struct Reference . . . . .	22
6.3.1	Detailed Description . . . . .	22
6.4	DSCase Struct Reference . . . . .	23
6.4.1	Detailed Description . . . . .	23
6.5	DSDesignSpace Struct Reference . . . . .	25
6.5.1	Detailed Description . . . . .	25
6.6	DSDictionary Struct Reference . . . . .	26
6.7	dsexpression Struct Reference . . . . .	27
6.7.1	Detailed Description . . . . .	27
6.8	DSGMASystem Struct Reference . . . . .	28
6.8.1	Detailed Description . . . . .	28
6.9	DSMatrix Struct Reference . . . . .	29
6.9.1	Detailed Description . . . . .	29
6.10	DSMatrixArray Struct Reference . . . . .	30
6.10.1	Detailed Description . . . . .	30
6.11	DSSSystem Struct Reference . . . . .	31
6.11.1	Detailed Description . . . . .	31
6.12	DSSStack Struct Reference . . . . .	32
6.13	DSSymbolicMatrix Struct Reference . . . . .	33
6.13.1	Detailed Description . . . . .	33
6.14	DSVariable Struct Reference . . . . .	34
6.14.1	Detailed Description . . . . .	34
6.15	DSVariablePool Struct Reference . . . . .	35
6.15.1	Detailed Description . . . . .	35
6.16	DSVertices Struct Reference . . . . .	36
6.16.1	Detailed Description . . . . .	36

6.17	<a href="#">expression_token Struct Reference</a>	37
6.17.1	<a href="#">Detailed Description</a>	37
6.18	<a href="#">matrix_token Struct Reference</a>	38
6.19	<a href="#">parse_expression_s Struct Reference</a>	39
6.19.1	<a href="#">Detailed Description</a>	39
6.20	<a href="#">parser_aux Struct Reference</a>	40
6.20.1	<a href="#">Detailed Description</a>	40
6.21	<a href="#">pthread_struct Struct Reference</a>	41
6.21.1	<a href="#">Detailed Description</a>	41
6.22	<a href="#">v_token_data Union Reference</a>	42
6.22.1	<a href="#">Detailed Description</a>	42
6.23	<a href="#">variable_token Struct Reference</a>	43
6.23.1	<a href="#">Detailed Description</a>	43
6.24	<a href="#">yy_buffer_state Struct Reference</a>	44
6.24.1	<a href="#">Field Documentation</a>	44
6.24.1.1	<a href="#">yy_bs_column</a>	44
6.24.1.2	<a href="#">yy_bs_lineno</a>	44
6.25	<a href="#">yy_trans_info Struct Reference</a>	45
6.26	<a href="#">yyguts_t Struct Reference</a>	46
6.26.1	<a href="#">Field Documentation</a>	46
6.26.1.1	<a href="#">yy_buffer_stack</a>	46
6.26.1.2	<a href="#">yy_buffer_stack_max</a>	46
6.26.1.3	<a href="#">yy_buffer_stack_top</a>	46
6.27	<a href="#">YYMINORTYPE Union Reference</a>	47
6.28	<a href="#">yyParser Struct Reference</a>	48
6.29	<a href="#">yyStackEntry Struct Reference</a>	49
<b>7</b>	<b>File Documentation</b>	<b>51</b>
7.1	<a href="#">DSDesignSpace.c File Reference</a>	51
7.1.1	<a href="#">Detailed Description</a>	53
7.2	<a href="#">DSDesignSpace.h File Reference</a>	54
7.2.1	<a href="#">Detailed Description</a>	55
7.3	<a href="#">DSDesignSpaceParallel.c File Reference</a>	56
7.3.1	<a href="#">Detailed Description</a>	56
7.4	<a href="#">DSDesignSpaceParallel.h File Reference</a>	58
7.4.1	<a href="#">Detailed Description</a>	58
7.5	<a href="#">DSErrors.c File Reference</a>	60

7.5.1	Detailed Description	60
7.5.2	Define Documentation	61
7.5.2.1	MSIZE	61
7.5.2.2	STACK_TRACE_NUM	61
7.5.3	Function Documentation	62
7.5.3.1	DSErrorFunction	62
7.6	DSErrors.h File Reference	63
7.6.1	Detailed Description	64
7.6.2	Define Documentation	65
7.6.2.1	DSError	65
7.6.3	Function Documentation	65
7.6.3.1	DSErrorFunction	65
7.7	DSExpression.c File Reference	66
7.7.1	Detailed Description	67
7.8	DSExpression.h File Reference	68
7.8.1	Detailed Description	69
7.9	DSExpressionTokenizerLex.c File Reference	70
7.9.1	Detailed Description	73
7.9.2	Define Documentation	73
7.9.2.1	YY_CURRENT_BUFFER	73
7.9.2.2	YY_DO_BEFORE_ACTION	74
7.9.2.3	YY_INPUT	74
7.9.2.4	yy_set_bol	74
7.9.2.5	yy_set_interactive	75
7.9.2.6	yyless	75
7.9.2.7	yyless	75
7.9.3	Function Documentation	75
7.9.3.1	DSExpressionFlex_flush_buffer	75
7.9.3.2	DSExpressionFlex_scan_buffer	76
7.9.3.3	DSExpressionFlex_scan_bytes	76
7.9.3.4	DSExpressionFlex_scan_string	76
7.9.3.5	DSExpressionFlexget_column	77
7.9.3.6	DSExpressionFlexget_extra	77
7.9.3.7	DSExpressionFlexget_in	77
7.9.3.8	DSExpressionFlexget_leng	77
7.9.3.9	DSExpressionFlexget_lineno	77

7.9.3.10	DSExpressionFlexget_out . . . . .	77
7.9.3.11	DSExpressionFlexget_text . . . . .	78
7.9.3.12	DSExpressionFlexpop_buffer_state . . . . .	78
7.9.3.13	DSExpressionFlexpush_buffer_state . . . . .	78
7.9.3.14	DSExpressionFlexset_column . . . . .	78
7.9.3.15	DSExpressionFlexset_extra . . . . .	78
7.9.3.16	DSExpressionFlexset_in . . . . .	79
7.9.3.17	DSExpressionFlexset_lineno . . . . .	79
7.10	DSGMASystem.c File Reference . . . . .	80
7.10.1	Detailed Description . . . . .	81
7.11	DSGMASystem.h File Reference . . . . .	82
7.11.1	Detailed Description . . . . .	82
7.12	DSGMASystemParsingAux.h File Reference . . . . .	84
7.12.1	Detailed Description . . . . .	85
7.12.2	Typedef Documentation . . . . .	85
7.12.2.1	gma_parseraux_t . . . . .	85
7.13	DSIO.c File Reference . . . . .	86
7.13.1	Detailed Description . . . . .	87
7.13.2	Function Documentation . . . . .	88
7.13.2.1	DSCaseStringInJSONFormat . . . . .	88
7.13.2.2	DSIOSetCaseJSONOptions . . . . .	88
7.13.2.3	DSIOSetErrorFile . . . . .	88
7.13.2.4	DSIOSetPostErrorFunction . . . . .	89
7.13.2.5	DSIOSetPostFatalErrorFunction . . . . .	89
7.13.2.6	DSIOSetPostWarningFunction . . . . .	89
7.13.2.7	DSIOSetPrintFunction . . . . .	89
7.13.2.8	DSIOSetSSystemJSONOptions . . . . .	90
7.13.2.9	DSMatrixArrayStringInJSONFormat . . . . .	90
7.13.2.10	DSMatrixStringInJSONFormat . . . . .	90
7.13.2.11	DSSSystemStringInJSONFormat . . . . .	90
7.13.2.12	DSVariablePoolStringInJSONFormat . . . . .	91
7.13.3	Variable Documentation . . . . .	91
7.13.3.1	DSCasePrintingOptions . . . . .	91
7.13.3.2	DSSSystemPrintingOptions . . . . .	91
7.14	DSIO.h File Reference . . . . .	92
7.14.1	Detailed Description . . . . .	94

7.14.2	Function Documentation	94
7.14.2.1	DSCaseStringInJSONFormat	94
7.14.2.2	DSIOSetCaseJSONOptions	94
7.14.2.3	DSIOSetErrorFile	95
7.14.2.4	DSIOSetPostErrorFunction	95
7.14.2.5	DSIOSetPostFatalErrorFunction	95
7.14.2.6	DSIOSetPostWarningFunction	96
7.14.2.7	DSIOSetPrintFunction	96
7.14.2.8	DSIOSetSSystemJSONOptions	96
7.14.2.9	DSMatrixArrayStringInJSONFormat	96
7.14.2.10	DSMatrixStringInJSONFormat	97
7.14.2.11	DSSSystemStringInJSONFormat	97
7.14.2.12	DSVariablePoolStringInJSONFormat	97
7.14.3	Variable Documentation	98
7.14.3.1	DSIOErrorFile	98
7.14.3.2	DSPostError	98
7.14.3.3	DSPostFatalError	98
7.14.3.4	DSPostWarning	98
7.14.3.5	DSPrintf	99
7.15	DSMatrix.h File Reference	100
7.15.1	Detailed Description	102
7.15.2	Function Documentation	103
7.15.2.1	DSMatrixAlloc	103
7.15.2.2	DSMatrixByAddingMatrix	103
7.15.2.3	DSMatrixByParsingString	104
7.15.2.4	DSMatrixBySubtractingMatrix	104
7.15.2.5	DSMatrixCalloc	104
7.15.2.6	DSMatrixCopy	105
7.15.2.7	DSMatrixDoubleValue	105
7.15.2.8	DSMatrixFree	105
7.15.2.9	DSMatrixIdentity	105
7.15.2.10	DSMatrixPLUDecomposition	106
7.15.2.11	DSMatrixRandomNumbers	106
7.15.2.12	DSMatrixSetDoubleValueAll	106
7.16	DSMatrix_gsl.c File Reference	107
7.16.1	Detailed Description	110



7.16.2	Function Documentation	110
7.16.2.1	DSMatrixAlloc	110
7.16.2.2	DSMatrixByAddingMatrix	110
7.16.2.3	DSMatrixByParsingString	111
7.16.2.4	DSMatrixBySubtractingMatrix	111
7.16.2.5	DSMatrixCalloc	112
7.16.2.6	DSMatrixCopy	112
7.16.2.7	DSMatrixDoubleValue	112
7.16.2.8	DSMatrixFree	113
7.16.2.9	DSMatrixIdentity	113
7.16.2.10	DSMatrixPLUDecomposition	113
7.16.2.11	DSMatrixRandomNumbers	113
7.16.2.12	DSMatrixSetDoubleValueAll	114
7.17	DSMatrixArray.c File Reference	115
7.17.1	Detailed Description	115
7.17.2	Function Documentation	116
7.17.2.1	DSMatrixArrayAddMatrix	116
7.17.2.2	DSMatrixArrayAlloc	116
7.17.2.3	DSMatrixArrayCopy	116
7.17.2.4	DSMatrixArrayFree	117
7.17.2.5	DSMatrixArrayMatrix	117
7.18	DSMatrixArray.h File Reference	118
7.18.1	Detailed Description	119
7.18.2	Function Documentation	119
7.18.2.1	DSMatrixArrayAddMatrix	119
7.18.2.2	DSMatrixArrayAlloc	119
7.18.2.3	DSMatrixArrayCopy	120
7.18.2.4	DSMatrixArrayFree	120
7.18.2.5	DSMatrixArrayMatrix	120
7.19	DSMatrixTokenizer.c File Reference	121
7.19.1	Detailed Description	121
7.20	DSMatrixTokenizer.h File Reference	122
7.20.1	Detailed Description	123
7.21	DSMatrixTokenizerLex.c File Reference	124
7.21.1	Detailed Description	127
7.21.2	Define Documentation	127

7.21.2.1	YY_CURRENT_BUFFER	127
7.21.2.2	YY_DO_BEFORE_ACTION	127
7.21.2.3	YY_INPUT	128
7.21.2.4	yy_set_bol	128
7.21.2.5	yy_set_interactive	128
7.21.2.6	yyless	129
7.21.2.7	yyless	129
7.21.3	Function Documentation	129
7.21.3.1	DSMatrixFlex_flush_buffer	129
7.21.3.2	DSMatrixFlex_scan_buffer	129
7.21.3.3	DSMatrixFlex_scan_bytes	130
7.21.3.4	DSMatrixFlex_scan_string	130
7.21.3.5	DSMatrixFlexget_column	130
7.21.3.6	DSMatrixFlexget_extra	131
7.21.3.7	DSMatrixFlexget_in	131
7.21.3.8	DSMatrixFlexget_leng	131
7.21.3.9	DSMatrixFlexget_lineno	131
7.21.3.10	DSMatrixFlexget_out	131
7.21.3.11	DSMatrixFlexget_text	131
7.21.3.12	DSMatrixFlexpop_buffer_state	132
7.21.3.13	DSMatrixFlexpush_buffer_state	132
7.21.3.14	DSMatrixFlexset_column	132
7.21.3.15	DSMatrixFlexset_extra	132
7.21.3.16	DSMatrixFlexset_in	132
7.21.3.17	DSMatrixFlexset_lineno	133
7.22	DSMemoryManager.c File Reference	134
7.22.1	Detailed Description	134
7.22.2	Function Documentation	135
7.22.2.1	DSSecureCalloc	135
7.22.2.2	DSSecureFree	135
7.22.2.3	DSSecureMalloc	136
7.22.2.4	DSSecureRealloc	136
7.23	DSMemoryManager.h File Reference	137
7.23.1	Detailed Description	137
7.23.2	Function Documentation	138
7.23.2.1	DSSecureCalloc	138

7.23.2.2	DSSecureFree	138
7.23.2.3	DSSecureMalloc	138
7.23.2.4	DSSecureRealloc	139
7.24	DSSystem.h File Reference	140
7.24.1	Detailed Description	141
7.25	DSStd.h File Reference	142
7.25.1	Detailed Description	142
7.26	DSSymbolicMatrix.h File Reference	144
7.26.1	Detailed Description	144
7.27	DSTypes.h File Reference	146
7.27.1	Detailed Description	147
7.27.2	Typedef Documentation	148
7.27.2.1	DSComplexMatrix	148
7.27.2.2	DSExpression	148
7.27.2.3	DSInternalDictionary	149
7.27.3	Enumeration Type Documentation	149
7.27.3.1	DSVariablePoolLock	149
7.28	DSVariable.c File Reference	150
7.28.1	Detailed Description	152
7.28.2	Function Documentation	152
7.28.2.1	DSVariableAlloc	152
7.28.2.2	DSVariableFree	152
7.28.2.3	DSVariablePoolAddVariable	153
7.28.2.4	DSVariablePoolAddVariableWithName	153
7.28.2.5	DSVariablePoolAlloc	153
7.28.2.6	DSVariablePoolCopy	154
7.28.2.7	DSVariablePoolFree	154
7.28.2.8	DSVariablePoolIsReadOnly	154
7.28.2.9	DSVariablePoolIsReadWrite	155
7.28.2.10	DSVariablePoolIsReadWriteAdd	155
7.28.2.11	DSVariablePoolNumberOfVariables	155
7.28.2.12	DSVariablePoolSetReadOnly	155
7.28.2.13	DSVariablePoolSetReadWrite	156
7.28.2.14	DSVariablePoolSetReadWriteAdd	156
7.28.2.15	DSVariableRelease	156
7.28.2.16	DSVariableRetain	157

7.29	DSVariable.h File Reference	158
7.29.1	Detailed Description	160
7.29.2	Function Documentation	160
7.29.2.1	DSVariableAlloc	160
7.29.2.2	DSVariableFree	160
7.29.2.3	DSVariablePoolAddVariable	161
7.29.2.4	DSVariablePoolAddVariableWithName	161
7.29.2.5	DSVariablePoolAlloc	161
7.29.2.6	DSVariablePoolCopy	162
7.29.2.7	DSVariablePoolFree	162
7.29.2.8	DSVariablePoolIsReadOnly	162
7.29.2.9	DSVariablePoolIsReadWrite	163
7.29.2.10	DSVariablePoolIsReadWriteAdd	163
7.29.2.11	DSVariablePoolNumberOfVariables	163
7.29.2.12	DSVariablePoolSetReadOnly	163
7.29.2.13	DSVariablePoolSetReadWrite	164
7.29.2.14	DSVariablePoolSetReadWriteAdd	164
7.29.2.15	DSVariableRelease	164
7.29.2.16	DSVariableRetain	165
7.30	DSVariableTokenizer.c File Reference	166
7.30.1	Detailed Description	166
7.31	DSVariableTokenizerLex.c File Reference	167
7.31.1	Detailed Description	170
7.31.2	Define Documentation	171
7.31.2.1	YY_CURRENT_BUFFER	171
7.31.2.2	YY_DO_BEFORE_ACTION	171
7.31.2.3	YY_INPUT	171
7.31.2.4	yy_set_bol	171
7.31.2.5	yy_set_interactive	172
7.31.2.6	yylless	172
7.31.2.7	yylless	172
7.31.3	Function Documentation	173
7.31.3.1	DSVariableFlex_create_buffer	173
7.31.3.2	DSVariableFlex_delete_buffer	173
7.31.3.3	DSVariableFlex_flush_buffer	173
7.31.3.4	DSVariableFlex_scan_buffer	173

7.31.3.5	DSVariableFlex_scan_bytes . . . . .	174
7.31.3.6	DSVariableFlex_scan_string . . . . .	174
7.31.3.7	DSVariableFlex_switch_to_buffer . . . . .	174
7.31.3.8	DSVariableFlexget_column . . . . .	174
7.31.3.9	DSVariableFlexget_extra . . . . .	175
7.31.3.10	DSVariableFlexget_in . . . . .	175
7.31.3.11	DSVariableFlexget_leng . . . . .	175
7.31.3.12	DSVariableFlexget_lineno . . . . .	175
7.31.3.13	DSVariableFlexget_out . . . . .	175
7.31.3.14	DSVariableFlexget_text . . . . .	175
7.31.3.15	DSVariableFlexpop_buffer_state . . . . .	176
7.31.3.16	DSVariableFlexpush_buffer_state . . . . .	176
7.31.3.17	DSVariableFlexrestart . . . . .	176
7.31.3.18	DSVariableFlexset_column . . . . .	176
7.31.3.19	DSVariableFlexset_extra . . . . .	176
7.31.3.20	DSVariableFlexset_in . . . . .	177
7.31.3.21	DSVariableFlexset_lineno . . . . .	177
7.31.4	Variable Documentation . . . . .	177
7.31.4.1	yy_current_state . . . . .	177



# Chapter 1

## Todo List

**File [DSErrors.c](#)** Implement locks when making the error strings.

**File [DSIO.h](#)** Define standard input and output file formats.  
Define criteria for warnings, errors and fatal errors.

**File [DSStd.h](#)** Add all previous functionality.  
Add vertex enumeration functionality.





# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

Macros to manipulate dictionary nodes. . . . .	9
Messages for DS Errors. . . . .	10
Actions for DS Errors. . . . .	12
DSGMAACCESSORS . . . . .	13
DS_IO_TAG_TYPES . . . . .	14
Options for JSON conversion of DSCase object. . . . .	15
Options for JSON conversion of DSSSystem object. . . . .	16
DSSSysACCESSORS . . . . .	17
Macros to manipulate variables. . . . .	18



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">_varDictionary</a> (Internal dictionary structure ) . . . . .	19
<a href="#">base_info</a> . . . . .	21
<a href="#">ds_parallelstack_t</a> (Stack object used by the worker threads ) . . . . .	22
<a href="#">DSCase</a> (Data type used to represent a case ) . . . . .	23
<a href="#">DSDesignSpace</a> (Data type used to represent a design space/ ) . . . . .	25
<a href="#">DSDictionary</a> . . . . .	26
<a href="#">dsexpression</a> (Data type representing mathematical expressions ) . . . . .	27
<a href="#">DSGMASystem</a> (Data type representing a GMA-System ) . . . . .	28
<a href="#">DSMatrix</a> (Data type representing a matrix ) . . . . .	29
<a href="#">DSMatrixArray</a> (Data type representing an array of matrices ) . . . . .	30
<a href="#">DSSSystem</a> (Data type representing an S-System ) . . . . .	31
<a href="#">DSSStack</a> . . . . .	32
<a href="#">DSSymbolicMatrix</a> (Data type representing a symbolic matrix ) . . . . .	33
<a href="#">DSVariable</a> (Basic variable structure containing name, value and NSString with special unicode characters for greek letters ) . . . . .	34
<a href="#">DSVariablePool</a> (User-level variable pool ) . . . . .	35
<a href="#">DSVertices</a> (Data type that contains vertices of an N-Dimensional object ) . . . . .	36
<a href="#">expression_token</a> (A data structure representing a token used when parsing strings for variable pools ) . . . . .	37
<a href="#">matrix_token</a> . . . . .	38
<a href="#">parse_expression_s</a> (Structure used when parsing a mathematical expression ) . . . . .	39
<a href="#">parser_aux</a> (Data type used to parse strings to GMA System ) . . . . .	40
<a href="#">pthread_struct</a> (Data structure passed to a pthread ) . . . . .	41
<a href="#">v_token_data</a> (Union containing the alternative values a struct <a href="#">variable_token</a> can take ) . . . . .	42
<a href="#">variable_token</a> (A data structure representing a token used when parsing strings for variable pools ) . . . . .	43
<a href="#">yy_buffer_state</a> . . . . .	44
<a href="#">yy_trans_info</a> . . . . .	45
<a href="#">yyguts_t</a> . . . . .	46
<a href="#">YYMINORTYPE</a> . . . . .	47
<a href="#">yyParser</a> . . . . .	48
<a href="#">yyStackEntry</a> . . . . .	49



# Chapter 4

## File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

<b>DSCase.h</b>	??
<a href="#">DSDesignSpace.c</a> (Implementation file with functions for dealing with Design Spaces )	51
<a href="#">DSDesignSpace.h</a> (Header file with functions for dealing with Design Spaces )	54
<a href="#">DSDesignSpaceParallel.c</a> (Implementation file with functions for dealing with parallel operations used by the design spaces )	56
<a href="#">DSDesignSpaceParallel.h</a> (Header file with functions for dealing with parallel operations used by the design spaces )	58
<b>DSDictionary.h</b>	??
<a href="#">DSErrors.c</a> (Implementation file with functions for error and exception handling )	60
<a href="#">DSErrors.h</a> (Header file with functions for error and exception handling )	63
<a href="#">DSExpression.c</a> (Implementation file with functions for dealing with mathematical expressions )	66
<a href="#">DSExpression.h</a> (Header file with functions for dealing with mathematical expressions )	68
<b>DSExpressionGrammar.h</b>	??
<b>DSExpressionTokenizer.h</b>	??
<a href="#">DSExpressionTokenizerLex.c</a> (Implementation file with functions for tokenizing matrices, generated by flex )	70
<a href="#">DSGMASystem.c</a> (Implementation file with functions for dealing with GMA Systems )	80
<a href="#">DSGMASystem.h</a> (Header file with functions for dealing with GMA Systems )	82
<b>DSGMASystemGrammar.h</b>	??
<a href="#">DSGMASystemParsingAux.h</a> (Implementation file with functions for dealing with the parsing of GMA Systems )	84
<a href="#">DSIO.c</a> (Implementation file with standard input and output functions )	86
<a href="#">DSIO.h</a> (Header file with standard input and output functions )	92
<a href="#">DSMatrix.h</a> (Header file with functions for dealing with matrices )	100
<a href="#">DSMatrix_gsl.c</a> (Implementation file with functions for dealing with matrices using the GNU Scientific Library (gsl) )	107
<a href="#">DSMatrixArray.c</a> (Implementation file with functions for dealing with matrix arrays )	115
<a href="#">DSMatrixArray.h</a> (Header file with functions for dealing with matrix arrays )	118
<a href="#">DSMatrixTokenizer.c</a> (Implementation file with functions for tokenizing with matrices )	121
<a href="#">DSMatrixTokenizer.h</a> (Header file with functions for tokenizing matrices )	122
<a href="#">DSMatrixTokenizerLex.c</a> (Implementation file with functions for tokenizing matrices, generated by flex )	124
<a href="#">DSMemoryManager.c</a> (Implementation file with functions for secure memory management )	134

<a href="#">DSMemoryManager.h</a> (Header file with functions for secure memory allocation ) . . . . .	137
<a href="#">DSSSystem.h</a> (Header file with functions for dealing with S-System ) . . . . .	140
<b>DSSSystemGrammar.h</b> . . . . .	??
<b>DSSStack.h</b> . . . . .	??
<a href="#">DSSStd.h</a> (Header file for the design space toolbox ) . . . . .	142
<b>DSSubcase.h</b> . . . . .	??
<a href="#">DSSymbolicMatrix.h</a> (Header file with functions for dealing with symbolic matrices ) . . . . .	144
<a href="#">DSTypes.h</a> (Header file with definitions for data types ) . . . . .	146
<a href="#">DSVariable.c</a> (Implementation file with functions the DSInternalDictionary object ) . . . . .	150
<a href="#">DSVariable.h</a> (Header file with functions for dealing with variables ) . . . . .	158
<b>DSVariableGrammar.h</b> . . . . .	??
<a href="#">DSVariableTokenizer.c</a> (Implementation file with functions for tokenizing with matrices ) . . . . .	166
<b>DSVariableTokenizer.h</b> . . . . .	??
<a href="#">DSVariableTokenizerLex.c</a> (Implementation file with functions for tokenizing matrices, generated by flex ) . . . . .	167
<b>DSVertices.h</b> . . . . .	??

# Chapter 5

## Module Documentation

### 5.1 Macros to manipulate dictionary nodes.

#### Defines

- `#define dsInternalDictionarySetValue(x, y) ((x != NULL) ? x->value = y : DSError(M_DS_WRONG ": Dictionary is NULL", A_DS_ERROR))`  
*Macro to set the value of a DSInternalDictionary node.*
- `#define dsInternalDictionaryValue(x) ((x != NULL) ? x->value : NULL)`  
*Macro to get the value of a dictionary node.*

#### 5.1.1 Detailed Description

The following macros are in place for portability and consistency. As the structure of the DSInternalDictionary is subject to changes, using these macros will make the dependent code less subject to errors.

#### 5.1.2 Define Documentation

##### 5.1.2.1 `#define dsInternalDictionarySetValue(x, y) ((x != NULL) ? x->value = y : DSError(M_DS_WRONG ": Dictionary is NULL", A_DS_ERROR))`

Macro to set the value of a DSInternalDictionary node.

This macro provides a consistent way for changing the value of a dictionary node, despite the internal structure of the data type. This macro is expanded to a simple assignment.

##### 5.1.2.2 `#define dsInternalDictionaryValue(x) ((x != NULL) ? x->value : NULL)`

Macro to get the value of a dictionary node.

This macro provides a consistent way for retrieving the value of a DSInternalDictionary node, despite changes in the internal structure of the data type.

## 5.2 Messages for DS Errors.

### Defines

- `#define M_DS_CASE_NULL M_DS_NULL ": Case is NULL"`  
*Error message indicating a NULL variable pool.*
- `#define M_DS_DICTIONARY_NULL M_DS_NULL ": Dictionary is NULL"`  
*Error message indicating a NULL variable pool.*
- `#define M_DS_NOFILE "File not found"`  
*Message for no file found.*
- `#define M_DS_NULL "NULL pointer"`  
*Message for NULL pointer.*
- `#define M_DS_NOFORMAT "Format not known"`  
*Message for unknown format.*
- `#define M_DS_EXISTS "Data already exists"`  
*Message for data already existing.*
- `#define M_DS_MALLOC "Memory alloc failed"`  
*Message for failure to allocate data.*
- `#define M_DS_NOT_IMPL "Functionality not implemented"`  
*Message for a feature not yet implemented.*
- `#define M_DS_MAT_NULL "Pointer to matrix is NULL"`  
*Message for a NULL DSMatrix pointer.*
- `#define M_DS_MAT_OUTOFBOUNDS "Row or column out of bounds"`  
*Message for a row or column exceeding matrix bounds.*
- `#define M_DS_MAT_NOINTERNAL "Matrix data is empty"`  
*Message for a NULL internal matrix structure.*
- `#define M_DS_SYM_MAT_NULL "Pointer to symbolic matrix is NULL"`  
*Message for a NULL DSMatrix pointer.*
- `#define M_DS_SYM_MAT_OUTOFBOUNDS "Row or column out of bounds"`  
*Message for a row or column exceeding matrix bounds.*
- `#define M_DS_SYM_MAT_NOINTERNAL "Matrix data is empty"`  
*Message for a NULL internal matrix structure.*
- `#define M_DS_VAR_NULL M_DS_NULL ": Variable Pool is NULL"`  
*Error message indicating a NULL variable pool.*
- `#define M_DS_VAR_LOCKED "DSVariablePool: Insufficient privileges"`  
*Error message indicating insufficient privileges to manipulate a variable pool.*



### 5.2.1 Detailed Description

Defined here are the generic messages used to report the appropriate errors. These are used with the different actions in the macro `DS_ERROR`. Other messages can be reported by literally writing them in instead of these messages in the `DSError` macro. Also, these messages can be modified by appending a literal string in the `DSError` macro.

#### See also

[Actions for DS Errors.](#)  
[DSError](#)

Messages for [DSCase](#) related errors is `M_DS_CASE_NULL`.

Messages for [DSVariable](#) related errors are `M_DS_VAR_NULL` and `M_DS_VAR_LOCKED`.

Messages for [DSMatrix](#) related errors are `M_DS_MAT_NULL`, `M_DS_MAT_OUTOFBOUNDS` and `M_DS_MAT_NOINTERNAL`.

## 5.3 Actions for DS Errors.

### Defines

- #define [A\\_DS\\_NOERROR](#) 0  
*Value for no error.*
- #define [A\\_DS\\_WARN](#) -1  
*Value for a warning.*
- #define [A\\_DS\\_ERROR](#) -2  
*Value for an error.*
- #define [A\\_DS\\_FATAL](#) -3  
*Value for a fatal error, kills program.*
- #define [A\\_DS\\_KILLNOW](#) A\_DS\_FATAL  
*DEPRECATED:*

### 5.3.1 Detailed Description

Defined here are the appropriate reactions to a specific error, an error can have different actions depending on the sensitivity of the region involved.

#### See also

[Messages for DS Errors.](#)  
DS\_ERROR

## 5.4 DSGMAACCESSORS

Internal GMA Accessor macros.

### Defines

- `#define DSGMAXi(x) ((x)->Xi)`
- `#define DSGMAXd(x) ((x)->Xd)`
- `#define DSGMAAlpha(x) ((x)->alpha)`
- `#define DSGMABeta(x) ((x)->beta)`
- `#define DSGMAGd(x) ((x)->Gd)`
- `#define DSGMAGi(x) ((x)->Gi)`
- `#define DSGMAHd(x) ((x)->Hd)`
- `#define DSGMAHi(x) ((x)->Hi)`
- `#define DSGMASignature(x) ((x)->signature)`

### 5.4.1 Detailed Description

Internal GMA Accessor macros. Used within [DSGMASystem.c](#) to access the data within a GMA data type. These macros are not to be used outside of this file, as they do not check the data for consistency and thus would not invoke the `DSError` function, making it harder to trace errors.

## 5.5 DS\_IO\_TAG\_TYPES

### Defines

- `#define DS_IO_TAG_TYPE_Matrix "\\DSMatrix\\"`
- `#define DS_IO_TAG_TYPE_MatrixArray "\\DSMatrixArray\\"`
- `#define DS_IO_TAG_TYPE_VariablePool "\\DSVariablePool\\"`
- `#define DS_IO_TAG_TYPE_Dictionary "\\DSDictionary\\"`
- `#define DS_IO_TAG_TYPE_SSystem "\\DSSystem\\"`
- `#define DS_IO_TAG_TYPE_Case "\\DSCase\\"`
- `#define DS_IO_TAG_TYPE_DesignSpace "\\DSDesignSpace\\"`

## 5.6 Options for JSON conversion of DSCase object.

### Defines

- #define [DS\\_CASE\\_JSON\\_NO\\_SSYSTEM](#) 1  
*Flag value indicating that the S-System information should not be included in the JSON string.*
- #define [DS\\_CASE\\_JSON\\_NO\\_CASE\\_SIGNATURE](#) 2  
*Flag value indicating that the case signature should not be included in the JSON string.*
- #define [DS\\_CASE\\_JSON\\_NO\\_CONDITIONS](#) 4  
*Flag value indicating that the conditions for validity should not be included in the JSON string.*

### 5.6.1 Detailed Description

Defined here are different options determining the information stored in a JSON string for a [DSCase](#) object. These options are passed to the `DSIOSetCaseJSONOptions` function. These options designate the value for a global flag variable

## 5.7 Options for JSON conversion of DSSSystem object.

### Defines

- #define [DS\\_SSYSTEM\\_JSON\\_NO\\_SOLUTION](#) 1  
*Flag value indicating that the S-System solution should not be included in the JSON string.*
- #define [DS\\_SSYSTEM\\_JSON\\_NO\\_SINGULAR](#) 2  
*Flag value indicating that the JSON string will not indicate if the S-System is singular.*

### 5.7.1 Detailed Description

Defined here are different options determining the information stored in a JSON string for a [DSSSystem](#) object. These options are passed to the `DSIOSetSSystemJSONOptions` function. These options designate the value for a global flag variable.

## 5.8 DSSSysACCESSORS

Internal S-System Accessor macros.

### Defines

- `#define DSSSysXi(x) ((x)->Xi)`
- `#define DSSSysXd(x) ((x)->Xd)`
- `#define DSSSysAlpha(x) ((x)->alpha)`
- `#define DSSSysBeta(x) ((x)->beta)`
- `#define DSSSysGd(x) ((x)->Gd)`
- `#define DSSSysGi(x) ((x)->Gi)`
- `#define DSSSysHd(x) ((x)->Hd)`
- `#define DSSSysHi(x) ((x)->Hi)`
- `#define DSSSysM(x) ((x)->M)`
- `#define DSSSysIsSingular(x) ((x)->isSingular)`
- `#define DSSSysShouldFreeXd(x) ((x)->shouldFreeXd)`
- `#define DSSSysShouldFreeXi(x) ((x)->shouldFreeXi)`

### 5.8.1 Detailed Description

Internal S-System Accessor macros. Used within DSSSystem.c to access the data within a S-System data type. These macros are not to be used outside of this file, as they do not check the data for consistency and thus would not invoke the DSError function, making it harder to trace errors.

## 5.9 Macros to manipulate variables.

### Defines

- `#define DSVariableSetValue(x, y) (((DSVariable*)(x))->value = (y))`  
Macro to set the value of a variable data structure.
- `#define DSVariableValue(x) (((x) != NULL) ? ((DSVariable*)(x))->value : NAN)`  
Macro to get the value of a variable data structure.
- `#define DSVariableName(x) (((DSVariable *)x)->name)`  
Macro to get the value of a variable data structure.

### 5.9.1 Detailed Description

The following macros are in place for portability and consistency. As the structure of the `DSVariable` is subject to change, due to the nature of early versions of the framework, using these macros will make the dependent code less subject to errors.

### 5.9.2 Define Documentation

#### 5.9.2.1 `#define DSVariableName(x) (((DSVariable *)x)->name)`

Macro to get the value of a variable data structure.

This macro provides a consistent way for retrieving the value of a variable, despite the internal structure of the data type.

#### 5.9.2.2 `#define DSVariableSetValue(x, y) (((DSVariable*)(x))->value = (y))`

Macro to set the value of a variable data structure.

This macro provides a consistent way for changing the value of a variable, despite the internal structure of the data type. This macro is expanded to a simple assignment.

#### 5.9.2.3 `#define DSVariableValue(x) (((x) != NULL) ? ((DSVariable*)(x))->value : NAN)`

Macro to get the value of a variable data structure.

This macro provides a consistent way for retrieving the value of a variable, despite the internal structure of the data type.



## Chapter 6

# Data Structure Documentation

### 6.1 `_varDictionary` Struct Reference

Internal dictionary structure.

```
#include <DSTypes.h>
```

Collaboration diagram for `_varDictionary`:

#### Data Fields

- char `current`  
*The current character in the dictionary.*
- struct `_varDictionary` \* `alt`  
*The alternative character in the dictionary.*
- struct `_varDictionary` \* `next`  
*The next character in the dictionary.*
- void \* `value`  
*The variable stored. Only when current is '\0'.*

#### 6.1.1 Detailed Description

Internal dictionary structure. Internal dictionary for fast variable querying. The structure of the dictionary uses an alternative path, where each character is checked in order at each position, if there is a match, the next position is consequently checked. The dictionary should never be manipulated manually, adding, retrieving and removing variables should be done through the accessory functions.

**See also**

[DSDictionary](#)

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

- [DSTypes.h](#)

## 6.2 base\_info Union Reference

### Data Fields

- char \* [name](#)  
*The string representing the name of the variable.*
- double [value](#)  
*The variable representing the value of a constant.*

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

- [DSGMASystemParsingAux.h](#)

## 6.3 ds\_parallelstack\_t Struct Reference

Stack object used by the worker threads.

```
#include <DSDesignSpaceParallel.h>
```

Collaboration diagram for ds\_parallelstack\_t:

### Data Fields

- DSUInteger \* [base](#)  
*The pointer to the array of DSUIntegers storing the case numbers.*
- DSUInteger \* [current](#)  
*A pointer to the top of the stack.*
- DSUInteger [count](#)  
*The number of elements in the stack.*
- DSUInteger [size](#)  
*The current size of the base array.*
- DSUInteger [nextIndex](#)  
*The index of the current case.*
- DSCase \*\* [cases](#)  
*The array of cases processed.*
- pthread\_mutex\_t [pushpop](#)  
*The mutex used when pushing and popping data from the stack.*

### 6.3.1 Detailed Description

Stack object used by the worker threads. This structure is a stack of case numbers indicating the DSCases that need to be processed, and each pthread\_t used for processing cases and determining validity (currently disabled due to the non re-entrant GLPK) must have access to a [ds\\_parallelstack\\_t](#).

#### Note

One stack should be created per thread, to avoid one thread blocking another during popping and pushing operations. A single stack could be used, as the parallel stacks are thread safe, and under some conditions might be more efficient as all the threads in the thread pool will remain active until all cases have been processed. Currently, the number of cases to be processed by a thread are determined prior to launching the threads, and each thread has an equal number of cases to process. If a thread has many invalid cases, it may finish all of its cases before the other threads, and thus it is possible for the system to make less use of multiple processors. To avoid this situation, more threads than processors can be used or a single shared stack could be used.

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

- [DSDesignSpaceParallel.h](#)

## 6.4 DSCase Struct Reference

Data type used to represent a case.

```
#include <DSTypes.h>
```

Collaboration diagram for DSCase:

### Data Fields

- `const DSVariablePool * Xd`  
*A pointer to the [DSVariablePool](#) with the dependent variables.*
- `const DSVariablePool * Xi`  
*A pointer to the [DSVariablePool](#) with the independent variables.*
- `DSSSystem * ssys`  
*The [DSSSystem](#) of the case.*
- `DSMatrix * Cd`  
*The condition matrix corresponding to the dependent variables.*
- `DSMatrix * Ci`  
*The condition matrix corresponding to the independent variables.*
- `DSMatrix * U`  
*The boundary matrix corresponding to the independent variables.*
- `DSMatrix * delta`  
*The condition matrix corresponding to the constants.*
- `DSMatrix * zeta`  
*The boundary matrix corresponding to the constants.*
- `DSUInteger caseNumber`  
*The case number used to identify the case.*
- `DSUInteger * signature`  
*The case signature indicating the dominant terms used to generate the case.*

### 6.4.1 Detailed Description

Data type used to represent a case. This data type has all the necessary information for a case in design space. It a pointer to the dependent and independent variables of the system, a pointer to the corresponding S-System, the Condition matrices and boundary matrices. It also has information about the case number and case signature.

#### Note

The case number is arbitrary, and can be generated by two algorithms to be either big endian or small endian. For compatibility with the current design space toolbox, big endian is the default.

The case is not responsible for freeing the Xd and Xi variables. If the case is generated from a design space, then the design space is responsible for freeing the Xi and Xd variable pools; otherwise the internal S-System is responsible for freeing this data.

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

- [DSTypes.h](#)

## 6.5 DSDesignSpace Struct Reference

Data type used to represent a design space/.

```
#include <DSTypes.h>
```

Collaboration diagram for DSDesignSpace:

### Data Fields

- [DSGMASystem](#) \* [gma](#)  
*The gma system of the design space.*
- const [DSVariablePool](#) \* [Xd](#)  
*A pointer to the [DSVariablePool](#) with the dependent variables.*
- const [DSVariablePool](#) \* [Xi](#)  
*A pointer to the [DSVariablePool](#) with the dependent variables.*
- [DSDictionary](#) \* [validCases](#)  
*[DSVariablePool](#) with case number that are valid.*
- DSUInteger [numberOfCases](#)  
*DSUInteger indicating the maximum number of cases in the design space.*
- [DSMatrix](#) \* [Cd](#)
- [DSMatrix](#) \* [Ci](#)
- [DSMatrix](#) \* [delta](#)  
*Condition matrices.*
- [DSDictionary](#) \* [subcases](#)  
*DSDesignSpaceStack containing design space objects with subcases.*

### 6.5.1 Detailed Description

Data type used to represent a design space/. The design space data structure is a convenience structure that automates the construction and analysis of cases, and manages the memory associated with these cases. This behavior can be avoided by working directly with the gma system of the designspace.

#### See also

[DSDesignSpace.h](#)  
[DSDesignSpace.c](#)

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

- [DSTypes.h](#)

## 6.6 DSDictionary Struct Reference

Collaboration diagram for DSDictionary:

### Data Fields

- [DSInternalDictionary](#) \* **internal**
- DSUInteger **count**
- char \*\* **names**

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

- [DSTypes.h](#)



## 6.7 dsexpression Struct Reference

Data type representing mathematical expressions.

```
#include <DSTypes.h>
```

Collaboration diagram for dsexpression:

### Data Fields

- union {  
    char **op\_code**  
    double **constant**  
    char \* **variable**  
        *A string with the name of the variable.*  
} **node**

*Union of data types potentially contained in the node.*

- int **type**  
*Integer specifying the type of node.*
- int **numberOfBranches**  
*Number of branches of children, relevant to operators and functions.*
- struct **dsexpression** \*\* **branches**  
*Array of expression nodes with children nodes.*

### 6.7.1 Detailed Description

Data type representing mathematical expressions. This data type is the internal representation of mathematical expressions. This data type is an Abstracts Syntax Tree with only three operators: '+', '\*' and '^'. All other operators ('-' and '/') are represented by a combination of the former operators. The DSExpression automatically groups constant values, and reserves the first branch of the multiplication and addition operator for constant values. These operators can have any number of branches. The '^' operator can have two, and only two, branches.

#### Note

Functions are handled as variables with a single argument

#### See also

[DSExpression.h](#)  
[DSExpression.c](#)

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

- [DSTypes.h](#)

## 6.8 DSGMASystem Struct Reference

Data type representing a GMA-System.

```
#include <DSTypes.h>
```

Collaboration diagram for DSGMASystem:

### Data Fields

- char \*\* **equations**
- [DSMatrix](#) \* **alpha**
- [DSMatrix](#) \* **beta**
- [DSMatrixArray](#) \* **Gd**
- [DSMatrixArray](#) \* **Gi**
- [DSMatrixArray](#) \* **Hd**
- [DSMatrixArray](#) \* **Hi**
- [DSVariablePool](#) \* **Xd**
- [DSVariablePool](#) \* **Xi**
- DSUInteger \* **signature**

### 6.8.1 Detailed Description

Data type representing a GMA-System. This data structure is a standard representation of an GMA using matrix notation. Here, the positive and negative terms are explicitly represented according to the Gs and Hs. Also, matrices are split up relating to either dependent and independent parameters. The GMA system uses an array of matrices to represent all the terms in all of the equations.

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

- [DSTypes.h](#)

## 6.9 DSMatrix Struct Reference

Data type representing a matrix.

```
#include <DSTypes.h>
```

### Data Fields

- void \* [mat](#)  
*The pointer to the internal representation of the matrix.*
- DSUInteger [rows](#)  
*A DSUInteger specifying the number of rows in the matrix.*
- DSUInteger [columns](#)  
*A DSUInteger specifying the number of columns in the matrix.*

### 6.9.1 Detailed Description

Data type representing a matrix. This data type is the front end of the matrix manipulation portion of the design space toolbox. Currently, the DST library uses the gsl library; however, it is designed to be used with different back-ends. In particular, the CLAPACK package should be considered, as it will offer better performance. Thus, the matrix API should be independent of implementation, and hence a new matrix library could be used if chosen.

#### See also

[DSMatrix.h](#)  
[DSMatrix.c](#)

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

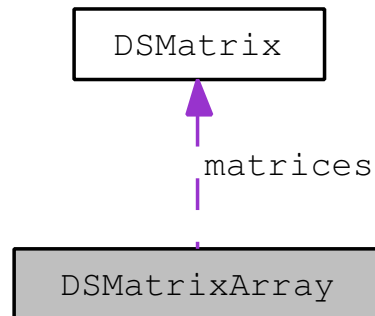
- [DSTypes.h](#)

## 6.10 DSMatrixArray Struct Reference

Data type representing an array of matrices.

```
#include <DSTypes.h>
```

Collaboration diagram for DSMatrixArray:



### Data Fields

- DSUInteger [numberOfMatrices](#)  
*A DSUInteger specifying the number of matrices in the array.*
- [DSMatrix](#) \*\* [matrices](#)  
*A pointer to the C-style array of matrices.*

### 6.10.1 Detailed Description

Data type representing an array of matrices. This data type is a utility data type that keeps track of arrays of matrices. This structure is used to represent three-dimensional matrices, as used internally by GMA's systems.

#### See also

[DSMatrixArray.h](#)  
[DSMatrixArray.c](#)

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

- [DSTypes.h](#)

## 6.11 DSSSystem Struct Reference

Data type representing an S-System.

```
#include <DSTypes.h>
```

Collaboration diagram for DSSSystem:

### Data Fields

- [DSMatrix](#) \* **alpha**
- [DSMatrix](#) \* **beta**
- [DSMatrix](#) \* **Gd**
- [DSMatrix](#) \* **Gi**
- [DSMatrix](#) \* **Hd**
- [DSMatrix](#) \* **Hi**
- [DSMatrix](#) \* **M**
- [DSVariablePool](#) \* **Xd**
- [DSVariablePool](#) \* **Xi**
- bool **isSingular**
- bool **shouldFreeXd**
- bool **shouldFreeXi**

### 6.11.1 Detailed Description

Data type representing an S-System. This data structure is a standard representation of an S-System using matrix notation. Here, the positive and negative terms are explicitly represented according to the Gs and Hs. Also, matrices are split up relating to either dependent and independent parameters.

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

- [DSTypes.h](#)

## 6.12 DSStack Struct Reference

### Data Fields

- void \*\* [base](#)  
*The pointer to the array of DSUIntegers storing the case numbers.*
- void \*\* [current](#)  
*A pointer to the top of the stack.*
- DSUInteger [count](#)  
*The number of elements in the stack.*
- DSUInteger [size](#)  
*The current size of the base array.*
- pthread\_mutex\_t [pushpop](#)  
*The mutex used when pushing and popping data from the stack.*

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

- [DSTypes.h](#)

## 6.13 DSSymbolicMatrix Struct Reference

Data type representing a symbolic matrix.

```
#include <DSTypes.h>
```

Collaboration diagram for DSSymbolicMatrix:

### Data Fields

- [DSExpression](#) \*\*\* **mat**
- DSUInteger **rows**
- DSUInteger **columns**

### 6.13.1 Detailed Description

Data type representing a symbolic matrix. This data type is the front end of the matrix manipulation portion of the design space toolbox involving symbolic data.. Currently, the DST library has a very limited manipulation of symbolic libraries, and is used exclusive to parse gma equations and design spaces. When performing any analysis of design space, the symbolic matrices are converted to numerical expressions.

#### See also

[DSMatrix.h](#)  
[DSMatrix.c](#)

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

- [DSTypes.h](#)

## 6.14 DSVariable Struct Reference

Basic variable structure containing name, value and NSString with special unicode characters for greek letters.

```
#include <DSTypes.h>
```

### Data Fields

- char \* [name](#)  
*Dynamically allocated name of the variable.*
- double [value](#)  
*Value of the variable.*
- DSUInteger [retainCount](#)  
*Retain counter for memory management.*

### 6.14.1 Detailed Description

Basic variable structure containing name, value and NSString with special unicode characters for greek letters. Structure that carries variable information. Internal to BSTVariables class and should not be created and/or freed manually and beyond the context of the BSTVariables class.

#### See also

[DSVariable.h](#)  
[DSVariable.c](#)

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

- [DSTypes.h](#)



## 6.15 DSVariablePool Struct Reference

User-level variable pool.

```
#include <DSTypes.h>
```

Collaboration diagram for DSVariablePool:

### Data Fields

- [DSDictionary](#) \* [dictionary](#)  
*The dictionary with the variables arranged.*
- DSUInteger [numberOfVariables](#)  
*Number of variables in the pool.*
- [DSVariable](#) \*\* [variables](#)  
*A C array with the variables stored.*
- [DSVariablePoolLock](#) [lock](#)  
*Indicates if the variable pool is read-only.*

### 6.15.1 Detailed Description

User-level variable pool. This data type keeps an internal dictionary structure of type struct [\\_varDictionary](#) to keep track of all the variables associated with a variable pool. This data type also records the number of variables in the dictionary and the order with which they were added.

#### See also

[struct \\_varDictionary](#)  
[DSVariable.h](#)  
[DSVariable.c](#)

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

- [DSTypes.h](#)

## 6.16 DSVertices Struct Reference

Data type that contains vertices of an N-Dimensional object.

```
#include <DSTypes.h>
```

### Data Fields

- double \*\* **vertices**
- DSUInteger **dimensions**
- DSUInteger **numberOfVertices**

### 6.16.1 Detailed Description

Data type that contains vertices of an N-Dimensional object. This data type is used for determining the region of validity of a case in design space. If the vertices represent a polygon, they can be ordered according to their clockwise position, starting by the right-most vertex in a XY plane.

#### See also

[DSVertices.h](#)  
[DSVertices.c](#)

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

- [DSTypes.h](#)

## 6.17 expression\_token Struct Reference

A data structure representing a token used when parsing strings for variable pools.

```
#include <DSExpressionTokenizer.h>
```

Collaboration diagram for expression\_token:

### Data Fields

- int [type](#)  
*The current token code.*
- union {
  - char \* [name](#)  
*Used for storing the name of a variable.*
  - double [value](#)  
*Used for storing the value of a constant.* } [data](#)  
  
*Union for holding either the name of a variable, or the value of a constant.*
- struct [expression\\_token](#) \* [next](#)  
*A pointer to the next token in the list.*

### 6.17.1 Detailed Description

A data structure representing a token used when parsing strings for variable pools. This structures follows the convention used with the struct [variable\\_token](#) and struct [matrix\\_token](#), representing an ordered list of tokens, as found by the tokenizers generated by the lex program.

#### See also

DSExpressionTokenizer()

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

- DSExpressionTokenizer.h

## 6.18 matrix\_token Struct Reference

Collaboration diagram for matrix\_token:

### Data Fields

- int **token**
- double **value**
- DSUInteger **row**
- DSUInteger **column**
- struct [matrix\\_token](#) \* **next**

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

- [DSMatrixTokenizer.h](#)

## 6.19 parse\_expression\_s Struct Reference

Structure used when parsing a mathematical expression.

```
#include <DSExpressionTokenizer.h>
```

Collaboration diagram for parse\_expression\_s:

### Data Fields

- [DSExpression](#) \* [root](#)

*The pointer to the DSExpression representing the root of the syntax tree.*

- bool [wasSuccessful](#)

*Indicates if the parsing was succesful.*

### 6.19.1 Detailed Description

Structure used when parsing a mathematical expression. This structure is used to parse a mathematical expression, it holds (1) the root of the abstract syntax tree and a flag indicating if any syntax errors were found.

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

- [DSExpressionTokenizer.h](#)

## 6.20 parser\_aux Struct Reference

Data type used to parse strings to GMA System.

```
#include <DSGMASystemParsingAux.h>
```

Collaboration diagram for parser\_aux:

### Data Structures

- union [base\\_info](#)

### Data Fields

- char [sign](#)  
*The sign of the term represented by the current node.*
- union [parser\\_aux::base\\_info](#) \* [bases](#)  
*Dynamically allocated array of bases, can be either variables or constants.*
- bool [succeeded](#)  
*A flag indicating if the parsing of the expression was succesful.*
- double \* [exponents](#)  
*A dynamically allocated array of exponents, must be constants.*
- DSUInteger [numberOfBases](#)  
*The number of base-exponents pairs in the term.*
- struct [parser\\_aux](#) \* [next](#)  
*A pointer to the next node, representing the next term in the equation.*

### 6.20.1 Detailed Description

Data type used to parse strings to GMA System. This data structure forms an organized list of terms, each with base exponent pairs that are then used to create the system matrices. This data structure is key for the parsing of GMA systems. Each node in the `gma_parser_aux_t` list represent a term in an expression in the order it was found, and each node points to the next term. Each expression, or equation, has it's own list of terms. If a base is a constant, then it should not have an exponent, and hence it's exponent is assigned a NAN value and this is used to indicate that the base is a constant.

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

- [DSGMASystemParsingAux.h](#)

## 6.21 pthread\_struct Struct Reference

Data structure passed to a pthread.

```
#include <DSDesignSpaceParallel.h>
```

Collaboration diagram for pthread\_struct:

### Data Fields

- [ds\\_parallelstack\\_t](#) \* **stack**
- [DSDesignSpace](#) \* **ds**
- FILE \* **file**

### 6.21.1 Detailed Description

Data structure passed to a pthread. This data structure has two fields, one is a pointer to a [ds\\_parallelstack\\_t](#) object; this stack contains a stack of case numbers to be processed in parallel. Each stack is not designed to be accessed concurrently, but should still be thread safe.

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

- [DSDesignSpaceParallel.h](#)

## 6.22 v\_token\_data Union Reference

Union containing the alternative values a struct [variable\\_token](#) can take.

```
#include <DSVariableTokenizer.h>
```

### Data Fields

- char \* **name**
- double **value**

### 6.22.1 Detailed Description

Union containing the alternative values a struct [variable\\_token](#) can take. The union can have either a string, used for the names of variables when an identifier is found; and a double value used when a value is found.

#### See also

struct [variable\\_token](#)

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

- DSVariableTokenizer.h



## 6.23 variable\_token Struct Reference

A data structure representing a token used when parsing strings for variable pools.

```
#include <DSVariableTokenizer.h>
```

Collaboration diagram for variable\_token:

### Data Fields

- int **type**
- union [v\\_token\\_data](#) **data**
- struct [variable\\_token](#) \* **next**

### 6.23.1 Detailed Description

A data structure representing a token used when parsing strings for variable pools.

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

- DSVariableTokenizer.h

## 6.24 yy\_buffer\_state Struct Reference

### Data Fields

- FILE \* **yy\_input\_file**
- char \* **yy\_ch\_buf**
- char \* **yy\_buf\_pos**
- yy\_size\_t **yy\_buf\_size**
- yy\_size\_t **yy\_n\_chars**
- int **yy\_is\_our\_buffer**
- int **yy\_is\_interactive**
- int **yy\_at\_bol**
- int [yy\\_bs\\_lineno](#)
- int [yy\\_bs\\_column](#)
- int **yy\_fill\_buffer**
- int **yy\_buffer\_status**

### 6.24.1 Field Documentation

#### 6.24.1.1 int yy\_bs\_column

The column count.

#### 6.24.1.2 int yy\_bs\_lineno

The line count.

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

- [DSExpressionTokenizerLex.c](#)
- [DSMatrixTokenizerLex.c](#)
- [DSVariableTokenizerLex.c](#)

## 6.25 yy\_trans\_info Struct Reference

### Data Fields

- flex\_int32\_t **yy\_verify**
- flex\_int32\_t **yy\_nxt**

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

- [DSExpressionTokenizerLex.c](#)
- [DSMatrixTokenizerLex.c](#)
- [DSVariableTokenizerLex.c](#)

## 6.26 yyguts\_t Struct Reference

Collaboration diagram for yyguts\_t:

### Data Fields

- YY\_EXTRA\_TYPE yyextra\_r
- FILE \* yyin\_r
- FILE \* yyout\_r
- size\_t yy\_buffer\_stack\_top
- size\_t yy\_buffer\_stack\_max
- YY\_BUFFER\_STATE \* yy\_buffer\_stack
- char yy\_hold\_char
- yy\_size\_t yy\_n\_chars
- yy\_size\_t yyleng\_r
- char \* yy\_c\_buf\_p
- int yy\_init
- int yy\_start
- int yy\_did\_buffer\_switch\_on\_eof
- int yy\_start\_stack\_ptr
- int yy\_start\_stack\_depth
- int \* yy\_start\_stack
- yy\_state\_type yy\_last\_accepting\_state
- char \* yy\_last\_accepting\_cpos
- int yylineno\_r
- int yy\_flex\_debug\_r
- char \* yytext\_r
- int yy\_more\_flag
- int yy\_more\_len

### 6.26.1 Field Documentation

#### 6.26.1.1 YY\_BUFFER\_STATE \* yy\_buffer\_stack

Stack as an array.

#### 6.26.1.2 size\_t yy\_buffer\_stack\_max

capacity of stack.

#### 6.26.1.3 size\_t yy\_buffer\_stack\_top

index of top of stack.

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

- [DSExpressionTokenizerLex.c](#)
- [DSMatrixTokenizerLex.c](#)
- [DSVariableTokenizerLex.c](#)

## 6.27 YYMINORTYPE Union Reference

### Data Fields

- int **yyinit**
- DSExpressionParserTOKENTYPE **yy0**
- DSGMASystemParserTOKENTYPE **yy0**
- DSSSystemParserTOKENTYPE **yy0**
- DSVariablePoolParserTOKENTYPE **yy0**

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

- DSExpressionGrammar.c
- DSGMASystemGrammar.c
- DSSSystemGrammar.c
- DSVariableGrammar.c

## 6.28 yyParser Struct Reference

Collaboration diagram for yyParser:

### Data Fields

- int **yyidx**
- int **yyerrcnt**
- DSExpressionParserARG\_SDECL [yyStackEntry](#) **yystack** [YYSTACKDEPTH]
- DSGMASystemParserARG\_SDECL int **yystksz**
- [yyStackEntry](#) \* **yystack**
- DSSSystemParserARG\_SDECL int **yystksz**
- DSVariablePoolParserARG\_SDECL int **yystksz**
- ParseARG\_SDECL int **yystksz**

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

- DSExpressionGrammar.c
- DSGMASystemGrammar.c
- DSSSystemGrammar.c
- DSVariableGrammar.c
- lempar.c

## 6.29 yyStackEntry Struct Reference

Collaboration diagram for yyStackEntry:

### Data Fields

- YYACTIONTYPE **stateno**
- YYCODETYPE **major**
- **YYMINORTYPE** **minor**

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

- DSExpressionGrammar.c
- DSGMASystemGrammar.c
- DSSSystemGrammar.c
- DSVariableGrammar.c
- lempar.c





# Chapter 7

## File Documentation

### 7.1 DSDesignSpace.c File Reference

Implementation file with functions for dealing with Design Spaces.

```
#include <stdio.h>
#include <string.h>
#include <pthread.h>
#include <glpk.h>
#include "DSMemoryManager.h"
#include "DSDesignSpace.h"
#include "DSMatrix.h"
#include "DSGMASystem.h"
#include "DSSSystem.h"
#include "DSCase.h"
#include "DSStack.h"
#include "DSTypes.h"
#include "DSErrors.h"
#include "DSSubcase.h"
```

Include dependency graph for DSDesignSpace.c: This graph shows which files directly or indirectly include this file:

#### Defines

- #define **\_\_DS\_MAC\_OS\_X\_\_**
- #define **DS\_PARALLEL\_DEFAULT\_THREADS** 3
- #define **DSDSGMA(x)** ((x)->gma)
- #define **DSDSNumCases(x)** ((x)->numberOfCases)
- #define **DSDSXd(x)** ((x)->Xd)
- #define **DSDSXi(x)** ((x)->Xi)

- `#define DSDSSubcases(x) ((x)->subcases)`
- `#define DSDSCi(x) ((x)->Ci)`
- `#define DSDSCd(x) ((x)->Cd)`
- `#define DSDSDelta(x) ((x)->delta)`
- `#define DSDSValidPool(x) ((x)->validCases)`

## Functions

- `DSDDesignSpace * DSDDesignSpaceAlloc (void)`
- `void DSDDesignSpaceFree (DSDDesignSpace *ds)`
- `DSDDesignSpace * DSDDesignSpaceByParsingStringList (const DSVariablePool *const Xd, const char *const string,...)`
- `DSDDesignSpace * DSDDesignSpaceByParsingStrings (const DSVariablePool *const Xd, char *const *const strings, const DSUInteger numberOfEquations)`
- `DSDDesignSpace * DSDDesignSpaceByParsingStringsWithXi (const DSVariablePool *const Xd, const DSVariablePool *const Xi, char *const *const strings, const DSUInteger numberOfEquations)`
- `void DSDDesignSpaceSetGMA (DSDDesignSpace *ds, DSGMASystem *gma)`
- `void DSDDesignSpaceAddConditions (DSDDesignSpace *ds, const DSMatrix *Cd, const DSMatrix *Ci, const DSMatrix *delta)`
- `const DSUInteger DSDDesignSpaceNumberOfEquations (const DSDDesignSpace *ds)`
- `DSExpression ** DSDDesignSpaceEquations (const DSDDesignSpace *ds)`
- `const DSUInteger DSDDesignSpaceNumberOfCases (const DSDDesignSpace *ds)`
- `const DSUInteger DSDDesignSpaceNumberOfValidCases (const DSDDesignSpace *ds)`
- `const DSUInteger * DSDDesignSpaceSignature (const DSDDesignSpace *ds)`
- `DSCase * DSDDesignSpaceCaseWithCaseNumber (const DSDDesignSpace *ds, const DSUInteger caseNumber)`
- `DSCase * DSDDesignSpaceCaseWithCaseSignature (const DSDDesignSpace *ds, const DSUInteger *signature)`
- `DSCase * DSDDesignSpaceCaseWithCaseSignatureList (const DSDDesignSpace *ds, const DSUInteger firstTerm,...)`
- `const bool DSDDesignSpaceCaseWithCaseNumberIsValid (const DSDDesignSpace *ds, const DSUInteger caseNumber)`
- `const bool DSDDesignSpaceCaseWithCaseSignatureIsValid (const DSDDesignSpace *ds, const DSUInteger *signature)`
- `const bool DSDDesignSpaceCaseWithCaseSignatureListIsValid (const DSDDesignSpace *ds, const DSUInteger firstTerm,...)`
- `const DSGMASystem * DSDDesignSpaceGMASystem (const DSDDesignSpace *ds)`
- `DSCase ** DSDDesignSpaceCalculateCases (DSDDesignSpace *ds, const DSUInteger numberOfCases, DSUInteger *cases)`
- `DSCase ** DSDDesignSpaceCalculateAllValidCases (DSDDesignSpace *ds)`
- `void DSDDesignSpaceCalculateUnderdeterminedCaseWithCaseNumber (DSDDesignSpace *ds, DSUInteger caseNumber)`
- `void DSDDesignSpaceCalculateUnderdeterminedCases (DSDDesignSpace *ds)`
- `void DSDDesignSpaceCalculateValidityOfCases (DSDDesignSpace *ds)`
- `void DSDDesignSpacePrint (const DSDDesignSpace *ds)`

### 7.1.1 Detailed Description

Implementation file with functions for dealing with Design Spaces. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.2 DSDesignSpace.h File Reference

Header file with functions for dealing with Design Spaces.

```
#include "DSTypes.h"
#include "DSErrors.h"
```

Include dependency graph for DSDesignSpace.h: This graph shows which files directly or indirectly include this file:

### Defines

- `#define M_DS_DESIGN_SPACE_NULL M_DS_NULL` ": Design Space is NULL"

### Functions

- `DSDesignSpace * DSDesignSpaceAlloc` (void)
- `void DSDesignSpaceFree` (`DSDesignSpace *ds`)
- `DSDesignSpace * DSDesignSpaceByParsingStringList` (const `DSVariablePool *const` Xd, const char \*const string,...)
- `DSDesignSpace * DSDesignSpaceByParsingStrings` (const `DSVariablePool *const` Xd, char \*const \*const strings, const `DSUInteger` numberOfEquations)
- `DSDesignSpace * DSDesignSpaceByParsingStringsWithXi` (const `DSVariablePool *const` Xd, const `DSVariablePool *const` Xi, char \*const \*const strings, const `DSUInteger` numberOfEquations)
- `void DSDesignSpaceSetGMA` (`DSDesignSpace *ds`, `DSGMASystem *gma`)
- `void DSDesignSpaceAddConditions` (`DSDesignSpace *ds`, const `DSMatrix *Cd`, const `DSMatrix *Ci`, const `DSMatrix *delta`)
- const `DSUInteger DSDesignSpaceNumberOfEquations` (const `DSDesignSpace *ds`)
- `DSExpression ** DSDesignSpaceEquations` (const `DSDesignSpace *ds`)
- const `DSUInteger DSDesignSpaceNumberOfValidCases` (const `DSDesignSpace *ds`)
- const `DSUInteger DSDesignSpaceNumberOfCases` (const `DSDesignSpace *ds`)
- const `DSUInteger * DSDesignSpaceSignature` (const `DSDesignSpace *ds`)
- `DSCase * DSDesignSpaceCaseWithCaseNumber` (const `DSDesignSpace *ds`, const `DSUInteger` caseNumber)
- `DSCase * DSDesignSpaceCaseWithCaseSignature` (const `DSDesignSpace *ds`, const `DSUInteger` \*signature)
- `DSCase * DSDesignSpaceCaseWithCaseSignatureList` (const `DSDesignSpace *ds`, const `DSUInteger` firstTerm,...)
- const bool `DSDesignSpaceCaseWithCaseNumberIsValid` (const `DSDesignSpace *ds`, const `DSUInteger` caseNumber)
- const bool `DSDesignSpaceCaseWithCaseSignatureIsValid` (const `DSDesignSpace *ds`, const `DSUInteger` \*signature)
- const bool `DSDesignSpaceCaseWithCaseSignatureListIsValid` (const `DSDesignSpace *ds`, const `DSUInteger` firstTerm,...)
- const `DSGMASystem * DSDesignSpaceGMASystem` (const `DSDesignSpace *ds`)
- `DSCase ** DSDesignSpaceCalculateCases` (`DSDesignSpace *ds`, const `DSUInteger` numberOfCase, `DSUInteger` \*cases)
- `DSCase ** DSDesignSpaceCalculateAllValidCases` (`DSDesignSpace *ds`)
- `void DSDesignSpaceCalculateUnderdeterminedCases` (`DSDesignSpace *ds`)
- `void DSDesignSpaceCalculateValidityOfCases` (`DSDesignSpace *ds`)
- `void DSDesignSpacePrint` (const `DSDesignSpace *ds`)

### 7.2.1 Detailed Description

Header file with functions for dealing with Design Spaces. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.3 DSDesignSpaceParallel.c File Reference

Implementation file with functions for dealing with parallel operations used by the design spaces.

```
#include <stdio.h>
#include <pthread.h>
#include <glpk.h>
#include "DSDesignSpaceParallel.h"
#include "DSErrors.h"
#include "DSMemoryManager.h"
#include "DSDesignSpace.h"
#include "DSGMASystem.h"
#include "DSSSystem.h"
#include "DSCase.h"
#include "DSMatrix.h"
#include <unistd.h>
```

Include dependency graph for DSDesignSpaceParallel.c:

### Defines

- #define **PARALLEL\_STACK\_SIZE\_INCREMENT** 5000

### Functions

- void **DSParallelInitMutexes** (void)
- [ds\\_parallelstack\\_t](#) \* **DSParallelStackAlloc** (void)
- void **DSParallelStackFree** ([ds\\_parallelstack\\_t](#) \*stack)
- void **DSParallelStackPush** ([ds\\_parallelstack\\_t](#) \*stack, const DSUInteger integer)
- const DSUInteger **DSParallelStackPop** ([ds\\_parallelstack\\_t](#) \*stack)
- void **DSParallelStackAddCase** ([ds\\_parallelstack\\_t](#) \*stack, [DSCase](#) \*aCase)
- void \* **DSParallelWorker** (void \*[pthread\\_struct](#))
- void \* **DSParallelWorkerCases** (void \*[pthread\\_struct](#))

### Variables

- [pthread\\_mutex\\_t](#) **workeradd**
- [pthread\\_mutex\\_t](#) **iomutex**

#### 7.3.1 Detailed Description

Implementation file with functions for dealing with parallel operations used by the design spaces. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.4 DSDesignSpaceParallel.h File Reference

Header file with functions for dealing with parallel operations used by the design spaces.

```
#include <pthread.h>
#include "DSTypes.h"
```

Include dependency graph for DSDesignSpaceParallel.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [ds\\_parallelstack\\_t](#)  
*Stack object used by the worker threads.*
- struct [pthread\\_struct](#)  
*Data structure passed to a pthread.*

### Functions

- void **DSPParallelInitMutexes** (void)
- [ds\\_parallelstack\\_t](#) \* **DSPParallelStackAlloc** (void)
- void **DSPParallelStackFree** ([ds\\_parallelstack\\_t](#) \*stack)
- void **DSPParallelStackPush** ([ds\\_parallelstack\\_t](#) \*stack, const DSUInteger number)
- const DSUInteger **DSPParallelStackPop** ([ds\\_parallelstack\\_t](#) \*stack)
- void **DSPParallelStackAddCase** ([ds\\_parallelstack\\_t](#) \*stack, [DSCase](#) \*aCase)
- void \* **DSPParallelWorkerCases** (void \*[pthread\\_struct](#))
- void \* **DSPParallelWorkerCasesSaveToDisk** (void \*[pthread\\_struct](#))
- void \* **DSPParallelWorkerValidity** (void \*[pthread\\_struct](#))

#### 7.4.1 Detailed Description

Header file with functions for dealing with parallel operations used by the design spaces. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.



### Date

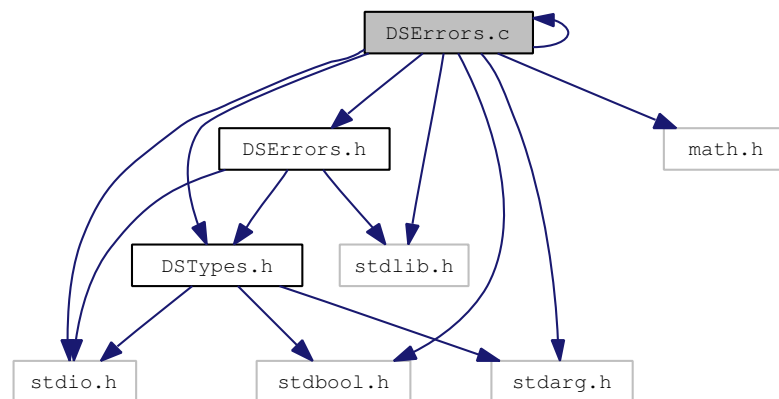
2011

## 7.5 DSErrors.c File Reference

Implementation file with functions for error and exception handling.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <execinfo.h>
#include "DSErrors.h"
#include "DSMemoryManager.h"
```

Include dependency graph for DSErrors.c:



### Defines

- `#define STACK\_TRACE\_NUM 10`  
*Maximum number of traces on the call stack.*
- `#define MSIZE 1500`  
*The maximum size of the error message string.*

### Functions

- `void DSErrorFunction (const char *M_DS_Message, char A_DS_ACTION, const char *FILEN, int LINE, const char *FUNC)`  
*Implicit error handling function. Called by DSError which automatically adds file and line arguments.*

#### 7.5.1 Detailed Description

Implementation file with functions for error and exception handling. This file specifies the design space standard for error handling. Contained here are the necessary macros and functions to report the errors

throughout the design space library. The DSErrorFunction allows different behaviors; the default behavior, errors are printed to the DSIOErrorFile, which is set to stderr by default. This behavior can be changed by setting changing DSPostWarning, DSPostError and DSPostFatalError function pointers.

**See also**

[DSIOErrorFile](#)  
[DSPostWarning](#)  
[DSPostError](#)  
[DSPostFatalError](#)

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

**Todo**

Implement locks when making the error strings.

## 7.5.2 Define Documentation

### 7.5.2.1 #define MSIZE 1500

The maximum size of the error message string.

This represents the maximum number of characters that an error string can contain. The error string is a statically allocated string.

### 7.5.2.2 #define STACK\_TRACE\_NUM 10

Maximum number of traces on the call stack.

This number represents the maximum number of traces on the call stack that the DSError function adds to the error string. The trace represents all the functions called up to the error.

### 7.5.3 Function Documentation

#### 7.5.3.1 void DSErrorFunction (const char \* *M\_DS\_Message*, char *A\_DS\_ACTION*, const char \* *FILEN*, int *LINE*, const char \* *FUNC*)

Implicit error handling function. Called by DSError which automatically adds file and line arguments.

This function is called implicitly when using the DSError macro. The DSError adds the FILE, LINE and FUNC arguments, to report the error/warning at the appropriate file, line and function.

#### Parameters

*M\_DS\_Message* A string containing the error message.

*A\_DS\_ACTION* A character representing an error code as described in A\_DS\_Actions.

*FILEN* A string with the name of the file where the error was reported.

*LINE* An integer with the line number in the file where the error was reported.

*FUNC* A string with the name of the function where the error was reported.

#### See also

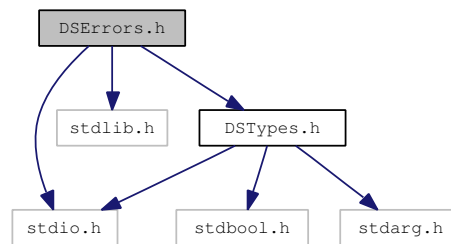
[DSError](#)  
[Actions for DS Errors.](#)

## 7.6 DSErrors.h File Reference

Header file with functions for error and exception handling.

```
#include <stdio.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSIO.h"
```

Include dependency graph for DSErrors.h:



This graph shows which files directly or indirectly include this file:

### Defines

- `#define M_DS_NOFILE` "File not found"  
*Message for no file found.*
- `#define M_DS_NULL` "NULL pointer"  
*Message for NULL pointer.*
- `#define M_DS_NOFORMAT` "Format not known"  
*Message for unknown format.*
- `#define M_DS_WRONG` "Inconsistent data"  
*Message for inconsistent data being used.*
- `#define M_DS_EXISTS` "Data already exists"  
*Message for data already existing.*
- `#define M_DS_NOTHREAD` "Thread not created"  
*Message for no thread created.*
- `#define M_DS_MALLOC` "Memory alloc failed"  
*Message for failure to allocate data.*
- `#define M_DS_NOT_IMPL` "Functionality not implemented"  
*Message for a feature not yet implemented.*
- `#define M_DS_PARSE` "Could not parse data"

*Message for an error during parsing.*

- `#define A_DS_NOERROR 0`  
*Value for no error.*
- `#define A_DS_WARN -1`  
*Value for a warning.*
- `#define A_DS_ERROR -2`  
*Value for an error.*
- `#define A_DS_FATAL -3`  
*Value for a fatal error, kills program.*
- `#define A_DS_KILLNOW A_DS_FATAL`  
*DEPRECATED:*
- `#define DSError(M_DS_Message, A_DS_Action) DSErrorFunction(M_DS_Message, A_DS_Action, __FILE__, __LINE__, __func__)`  
*Error reporting macro.*

## Functions

- `void DSErrorFunction (const char *M_DS_Message, char A_DS_ACTION, const char *FILEN, int LINE, const char *FUNC)`  
*Implicit error handling function. Called by DSError which automatically adds file and line arguments.*

### 7.6.1 Detailed Description

Header file with functions for error and exception handling. This file specifies the design space standard for error handling. Contained here are the necessary macros and functions to successfully report the errors throughout the design space library.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

**Date**

2011

**7.6.2 Define Documentation****7.6.2.1 #define DSError(M\_DS\_Message, A\_DS\_Action) DSErrorFunction(M\_DS\_Message, A\_DS\_Action, \_\_FILE\_\_, \_\_LINE\_\_, \_\_func\_\_)**

Error reporting macro.

Definition of the error reporting macro used within the DesignSpace C toolbox, this is a define which takes a string, which may be a standard message, and an action and reports it via the standard warning and error posting functions in the standard IO functions. A default behavior of the DSError macro posts warning and errors to stderr, while a fatal error posts the error to stderr and aborts the program.

**See also**

[DSPostWarning](#)  
[DSPostError](#)  
[DSPostFatalError](#)  
[Messages for DS Errors.](#)  
[Actions for DS Errors.](#)

**7.6.3 Function Documentation****7.6.3.1 void DSErrorFunction (const char \* *M\_DS\_Message*, char *A\_DS\_ACTION*, const char \* *FILEN*, int *LINE*, const char \* *FUNC*)**

Implicit error handling function. Called by DSError which automatically adds file and line arguments.

This function is called implicitly when using the DSError macro. The DSError adds the FILE, LINE and FUNC arguments, to report the error/warning at the appropriate file, line and function.

**Parameters**

*M\_DS\_Message* A string containing the error message.  
*A\_DS\_ACTION* A character representing an error code as described in A\_DS\_Actions.  
*FILEN* A string with the name of the file where the error was reported.  
*LINE* An integer with the line number in the file where the error was reported.  
*FUNC* A string with the name of the function where the error was reported.

**See also**

[DSError](#)  
[Actions for DS Errors.](#)

## 7.7 DSEexpression.c File Reference

Implementation file with functions for dealing with mathematical expressions.

```
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "DSEerrors.h"
#include "DSMemoryManager.h"
#include "DSVariable.h"
#include "DSEexpression.h"
#include "DSEexpressionTokenizer.h"
#include "DSTypes.h"
```

Include dependency graph for DSEexpression.c: This graph shows which files directly or indirectly include this file:

### Defines

- `#define DS_EXPRESSION_CONSTANT_BRANCH 0`
- `#define DS_EXPRESSION_STRING_INIT_LENGTH 1000`
- `#define ds_function_index_log 0`
- `#define ds_function_index_ln 1`
- `#define ds_function_index_log10 2`
- `#define ds_function_index_cos 3`
- `#define ds_function_index_sin 4`

### Functions

- `DSEexpression * DSEexpressionAllocWithOperator` (const char op\_code)
- `DSEexpression * DSEexpressionAllocWithConstant` (const double value)
- `DSEexpression * DSEexpressionAllocWithVariableName` (const char \*name)
- void `DSEexpressionFree` (DSEexpression \*root)
- `DSEexpression * DSEexpressionCopy` (const DSEexpression \*expression)
- `DSEexpression * DSEexpressionByParsingString` (const char \*string)
- `DSEexpression * DSEexpressionAddExpressions` (DSEexpression \*lvalue, DSEexpression \*rvalue)
- void `DSEexpressionAddBranch` (DSEexpression \*expression, DSEexpression \*branch)
- double `DSEexpressionEvaluateWithVariablePool` (const DSEexpression \*expression, const DSVariablePool \*pool)
- char \* `DSEexpressionAsString` (const DSEexpression \*expression)
- char \* `DSEexpressionAsTroffString` (const DSEexpression \*expression)
- void `DSEexpressionPrint` (const DSEexpression \*expression)



### 7.7.1 Detailed Description

Implementation file with functions for dealing with mathematical expressions. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.8 DSExpression.h File Reference

Header file with functions for dealing with mathematical expressions.

```
#include "DSTypes.h"
```

Include dependency graph for DSExpression.h: This graph shows which files directly or indirectly include this file:

### Defines

- `#define DS_EXPRESSION_TYPE_UNDEFINED 0`
- `#define DS_EXPRESSION_TYPE_OPERATOR 1`
- `#define DS_EXPRESSION_TYPE_CONSTANT 2`
- `#define DS_EXPRESSION_TYPE_VARIABLE 3`
- `#define DS_EXPRESSION_TYPE_FUNCTION 4`
- `#define DSExpressionSetOperator(x, y) ((x->node.op_code) = y, (x->type = DS_EXPRESSION_TYPE_OPERATOR))`
- `#define DSExpressionSetVariable(x, y) ((x->node.variable) = y, (x->type = DS_EXPRESSION_TYPE_VARIABLE))`
- `#define DSExpressionSetConstant(x, y) ((x->node.constant) = y, (x->type = DS_EXPRESSION_TYPE_CONSTANT))`
- `#define DSExpressionType(x) (x->type)`
- `#define DSExpressionNumberOfBranches(x) (x->numberOfBranches)`
- `#define DSExpressionBranchAtIndex(x, y) ((y < DSExpressionNumberOfBranches(x)) ? x->branches[y] : NULL)`
- `#define DSExpressionOperator(x) ((x->type == DS_EXPRESSION_TYPE_OPERATOR) ? x->node.op_code : '?')`
- `#define DSExpressionVariable(x) ((x->type == DS_EXPRESSION_TYPE_VARIABLE || x->type == DS_EXPRESSION_TYPE_FUNCTION) ? x->node.variable : NULL)`
- `#define DSExpressionConstant(x) ((x->type == DS_EXPRESSION_TYPE_CONSTANT) ? x->node.constant : NAN)`

### Functions

- `DSExpression * DSExpressionAllocWithOperator (const char op_code)`
- `DSExpression * DSExpressionAllocWithConstant (const double value)`
- `DSExpression * DSExpressionAllocWithVariableName (const char *name)`
- `void DSExpressionFree (DSExpression *root)`
- `DSExpression * DSExpressionCopy (const DSExpression *expression)`
- `DSExpression * DSExpressionByParsingString (const char *string)`
- `DSExpression * DSExpressionAddExpressions (DSExpression *lvalue, DSExpression *rvalue)`
- `double DSExpressionEvaluateWithVariablePool (const DSExpression *expression, const DSVariablePool *pool)`
- `char * DSExpressionAsString (const DSExpression *expression)`
- `char * DSExpressionAsTroffString (const DSExpression *expression)`
- `void DSExpressionPrint (const DSExpression *expression)`

### 7.8.1 Detailed Description

Header file with functions for dealing with mathematical expressions. The mathematical expressions are converted into a form similar to the model used in MUPAD. Internally, only three operators are used: '+', '\*' and '^'. The '-' operator is converted, such that  $A-B$  would actually be  $A+B*(-1)$  and the '/' operator is converted such that  $A/B$  would actually be  $A*B^{-1}$ . The '\*' and '+' operators must have at least two branches, but may have any number of branches. The first branch for these operators is reserved for constant values, such that  $a+b$  is actually  $0+a+b$ , and  $a*b$  is actually  $1*a*b$ . This canonical form is used to speed up the processing of mathematical expressions when converting them to matrices for the GMA and SSystem. The '^' must have only two branches.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.9 DSExpressionTokenizerLex.c File Reference

Implementation file with functions for tokenizing matrices, generated by flex.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSMemoryManager.h"
#include "DSExpression.h"
#include "DSExpressionTokenizer.h"
#include <unistd.h>
```

Include dependency graph for DSExpressionTokenizerLex.c:

### Data Structures

- struct [yy\\_buffer\\_state](#)
- struct [yy\\_trans\\_info](#)
- struct [yyguts\\_t](#)

### Defines

- #define **YY\_INT\_ALIGNED** short int
- #define **FLEX\_SCANNER**
- #define **YY\_FLEX\_MAJOR\_VERSION** 2
- #define **YY\_FLEX\_MINOR\_VERSION** 5
- #define **YY\_FLEX\_SUBMINOR\_VERSION** 35
- #define **FLEX\_BETA**
- #define **INT16\_MIN** (-32767-1)
- #define **INT32\_MIN** (-2147483647-1)
- #define **INT8\_MAX** (127)
- #define **INT16\_MAX** (32767)
- #define **INT32\_MAX** (2147483647)
- #define **UINT8\_MAX** (255U)
- #define **UINT16\_MAX** (65535U)
- #define **UINT32\_MAX** (4294967295U)
- #define **yyconst**
- #define **YY\_NULL** 0
- #define **YY\_SC\_TO\_UI**(c) ((unsigned int) (unsigned char) c)
- #define **YY\_TYPEDEF\_YY\_SCANNER\_T**
- #define **yyin** yyg->yyin\_r
- #define **yyout** yyg->yyout\_r
- #define **yyextra** yyg->yyextra\_r
- #define **yyleng** yyg->yyleng\_r
- #define **yytext** yyg->yytext\_r

- `#define yylineno (YY_CURRENT_BUFFER_LVALUE->yy_bs_lineno)`
- `#define yycolumn (YY_CURRENT_BUFFER_LVALUE->yy_bs_column)`
- `#define yy_flex_debug yyg->yy_flex_debug_r`
- `#define BEGIN yyg->yy_start = 1 + 2 *`
- `#define YY_START ((yyg->yy_start - 1) / 2)`
- `#define YYSTATE YY_START`
- `#define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)`
- `#define YY_NEW_FILE DSExpressionFlexrestart(yyin ,yyscanner )`
- `#define YY_END_OF_BUFFER_CHAR 0`
- `#define YY_BUF_SIZE 16384`
- `#define YY_STATE_BUF_SIZE ((YY_BUF_SIZE + 2) * sizeof(yy_state_type))`
- `#define YY_TYPEDEF_YY_BUFFER_STATE`
- `#define YY_TYPEDEF_YY_SIZE_T`
- `#define EOB_ACT_CONTINUE_SCAN 0`
- `#define EOB_ACT_END_OF_FILE 1`
- `#define EOB_ACT_LAST_MATCH 2`
- `#define YY_LESS_LINENO(n)`
- `#define yyless(n)`
- `#define unput(c) yyunput( c, yyg->ytext_ptr , yyscanner )`
- `#define YY_STRUCT_YY_BUFFER_STATE`
- `#define YY_BUFFER_NEW 0`
- `#define YY_BUFFER_NORMAL 1`
- `#define YY_BUFFER_EOF_PENDING 2`
- `#define YY_CURRENT_BUFFER`
- `#define YY_CURRENT_BUFFER_LVALUE yyg->yy_buffer_stack[yyg->yy_buffer_stack_top]`
- `#define YY_FLUSH_BUFFER DSExpressionFlex_flush_buffer(YY_CURRENT_BUFFER ,yyscanner)`
- `#define yy_new_buffer DSExpressionFlex_create_buffer`
- `#define yy_set_interactive(is_interactive)`
- `#define yy_set_bol(at_bol)`
- `#define YY_AT_BOL() (YY_CURRENT_BUFFER_LVALUE->yy_at_bol)`
- `#define ytext_ptr ytext_r`
- `#define YY_DO_BEFORE_ACTION`
- `#define YY_NUM_RULES 13`
- `#define YY_END_OF_BUFFER 14`
- `#define REJECT reject_used_but_not_detected`
- `#define yymore() ymore_used_but_not_detected`
- `#define YY_MORE_ADJ 0`
- `#define YY_RESTORE_YY_MORE_OFFSET`
- `#define malloc(x) DSSecureMalloc(x)`
- `#define calloc(x, y) DSSecureCalloc(x, y)`
- `#define realloc(x, y) DSSecureRealloc(x, y)`
- `#define INITIAL 0`
- `#define YY_EXTRA_TYPE struct expression\_token *`
- `#define YY_READ_BUF_SIZE 8192`
- `#define ECHO fwrite( ytext, yyleng, 1, yyout )`
- `#define YY_INPUT(buf, result, max_size)`
- `#define yyterminate() return YY_NULL`
- `#define YY_START_STACK_INCR 25`

- `#define YY_FATAL_ERROR(msg) yy_fatal_error( msg , yyscanner)`
- `#define YY_DECL_IS_OURS 1`
- `#define YY_DECL int DSExpressionFlexlex (yyscan_t yyscanner)`
- `#define YY_USER_ACTION`
- `#define YY_BREAK break;`
- `#define YY_RULE_SETUP YY_USER_ACTION`
- `#define YY_EXIT_FAILURE 2`
- `#define yyless(n)`
- `#define YYTABLES_NAME "yytables"`

## Typedefs

- `typedef signed char flex_int8_t`
- `typedef short int flex_int16_t`
- `typedef int flex_int32_t`
- `typedef unsigned char flex_uint8_t`
- `typedef unsigned short int flex_uint16_t`
- `typedef unsigned int flex_uint32_t`
- `typedef void * yyscan_t`
- `typedef struct yy_buffer_state * YY_BUFFER_STATE`
- `typedef size_t yy_size_t`
- `typedef unsigned char YY_CHAR`
- `typedef int yy_state_type`

## Functions

- `void DSExpressionFlexrestart (FILE *input_file, yyscan_t yyscanner)`
- `void DSExpressionFlex_switch_to_buffer (YY_BUFFER_STATE new_buffer, yyscan_t yyscanner)`
- `YY_BUFFER_STATE DSExpressionFlex_create_buffer (FILE *file, int size, yyscan_t yyscanner)`
- `void DSExpressionFlex_delete_buffer (YY_BUFFER_STATE b, yyscan_t yyscanner)`
- `void DSExpressionFlex_flush_buffer (YY_BUFFER_STATE b, yyscan_t yyscanner)`
- `void DSExpressionFlexpush_buffer_state (YY_BUFFER_STATE new_buffer, yyscan_t yyscanner)`
- `void DSExpressionFlexpop_buffer_state (yyscan_t yyscanner)`
- `YY_BUFFER_STATE DSExpressionFlex_scan_buffer (char *base, yy_size_t size, yyscan_t yyscanner)`
- `YY_BUFFER_STATE DSExpressionFlex_scan_string (yyconst char *yy_str, yyscan_t yyscanner)`
- `YY_BUFFER_STATE DSExpressionFlex_scan_bytes (yyconst char *bytes, yy_size_t len, yyscan_t yyscanner)`
- `void * DSExpressionFlexalloc (yy_size_t, yyscan_t yyscanner)`
- `void * DSExpressionFlexrealloc (void *, yy_size_t, yyscan_t yyscanner)`
- `void DSExpressionFlexfree (void *, yyscan_t yyscanner)`
- `int DSExpressionFlexlex_init (yyscan_t *scanner)`
- `int DSExpressionFlexlex_init_extra (YY_EXTRA_TYPE user_defined, yyscan_t *scanner)`
- `int DSExpressionFlexlex_destroy (yyscan_t yyscanner)`
- `int DSExpressionFlexget_debug (yyscan_t yyscanner)`
- `void DSExpressionFlexset_debug (int debug_flag, yyscan_t yyscanner)`
- `YY_EXTRA_TYPE DSExpressionFlexget_extra (yyscan_t yyscanner)`

- void [DSExpressionFlexset\\_extra](#) (YY\_EXTRA\_TYPE user\_defined, yyscan\_t yyscanner)
- FILE \* [DSExpressionFlexget\\_in](#) (yyscan\_t yyscanner)
- void [DSExpressionFlexset\\_in](#) (FILE \*in\_str, yyscan\_t yyscanner)
- FILE \* [DSExpressionFlexget\\_out](#) (yyscan\_t yyscanner)
- void [DSExpressionFlexset\\_out](#) (FILE \*out\_str, yyscan\_t yyscanner)
- yy\_size\_t [DSExpressionFlexget\\_leng](#) (yyscan\_t yyscanner)
- char \* [DSExpressionFlexget\\_text](#) (yyscan\_t yyscanner)
- int [DSExpressionFlexget\\_lineno](#) (yyscan\_t yyscanner)
- void [DSExpressionFlexset\\_lineno](#) (int line\_number, yyscan\_t yyscanner)
- int [DSExpressionFlexwrap](#) (yyscan\_t yyscanner)
- int [DSExpressionFlexlex](#) (yyscan\_t yyscanner)
- int [DSExpressionFlexget\\_column](#) (yyscan\_t yyscanner)
- void [DSExpressionFlexset\\_column](#) (int column\_no, yyscan\_t yyscanner)
- struct [expression\\_token](#) \* [DSExpressionTokenizeString](#) (const char \*string)

## 7.9.1 Detailed Description

Implementation file with functions for tokenizing matrices, generated by flex. This file was generated directly by the flex program, and is the source code responsible for matrix tokenization. This file was generated by flex, according to a specification written by Jason Lomnitz. To generate this file, the following command must be executed: "flex -t DSExpressionGrammar.l > DSExpressionTokenizerLex.c".

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

### Author

Jason Lomnitz.

### Date

2011

## 7.9.2 Define Documentation

### 7.9.2.1 #define YY\_CURRENT\_BUFFER

#### Value:

```
( yyg->yy_buffer_stack \
    ? yyg->yy_buffer_stack[yyg->yy_buffer_stack_top] \
    : NULL)
```

### 7.9.2.2 #define YY\_DO\_BEFORE\_ACTION

**Value:**

```
yyg->yytext_ptr = yy_bp; \
    yylen = (size_t) (yy_cp - yy_bp); \
    yyg->yy_hold_char = *yy_cp; \
    *yy_cp = '\0'; \
    yyg->yy_c_buf_p = yy_cp;
```

### 7.9.2.3 #define YY\_INPUT(buf, result, max\_size)

**Value:**

```
if ( YY_CURRENT_BUFFER_LVALUE->yy_is_interactive ) \
    { \
        int c = '*'; \
        yy_size_t n; \
        for ( n = 0; n < max_size && \
              (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
            buf[n] = (char) c; \
        if ( c == '\n' ) \
            buf[n++] = (char) c; \
        if ( c == EOF && ferror( yyin ) ) \
            YY_FATAL_ERROR( "input in flex scanner failed" ); \
        result = n; \
    } \
else \
    { \
        errno=0; \
        while ( (result = fread(buf, 1, max_size, yyin))==0 && ferror(yyi
n)) \
        { \
            if( errno != EINTR) \
            { \
                YY_FATAL_ERROR( "input in flex scanner failed" ); \
                break; \
            } \
            errno=0; \
            clearerr(yyin); \
        } \
    } \
\
```

### 7.9.2.4 #define yy\_setBol(atBol)

**Value:**

```
{ \
    if ( ! YY_CURRENT_BUFFER ){ \
        DSExpressionFlexensure_buffer_stack (yyscanner); \
        YY_CURRENT_BUFFER_LVALUE = \
            DSExpressionFlex_create_buffer(yyin,YY_BUF_SIZE ,yyscanner); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_atBol = atBol; \
}
```



### 7.9.2.5 #define yy\_set\_interactive(is\_interactive)

**Value:**

```
{ \
    if ( ! YY_CURRENT_BUFFER ){ \
        DSEexpressionFlexensure_buffer_stack (yyscanner); \
        YY_CURRENT_BUFFER_LVALUE = \
            DSEexpressionFlex_create_buffer(yyin,YY_BUF_SIZE ,yyscanner); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_is_interactive = is_interactive; \
}
```

### 7.9.2.6 #define yyless(n)

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        yytext[yytextlen] = yyg->yy_hold_char; \
        yyg->yy_c_buf_p = yytext + yyless_macro_arg; \
        yyg->yy_hold_char = *yyg->yy_c_buf_p; \
        *yyg->yy_c_buf_p = '\0'; \
        yytextlen = yyless_macro_arg; \
    } \
    while ( 0 )
```

### 7.9.2.7 #define yyless(n)

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        *yy_cp = yyg->yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yyg->yy_c_buf_p = yy_cp = yy_bp + yyless_macro_arg - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
    while ( 0 )
```

## 7.9.3 Function Documentation

### 7.9.3.1 void DSEexpressionFlex\_flush\_buffer (YY\_BUFFER\_STATE *b*, *yyscanner*)

Discard all buffered characters. On the next scan, YY\_INPUT will be called.

#### Parameters

*b* the buffer state to be flushed, usually YY\_CURRENT\_BUFFER.

*yyscanner* The scanner object.

### 7.9.3.2 YY\_BUFFER\_STATE DSExpressionFlex\_scan\_buffer (char \* *base*, yy\_size\_t *size*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan directly from a user-specified character buffer.

#### Parameters

*base* the character buffer  
*size* the size in bytes of the character buffer  
*yyscanner* The scanner object.

#### Returns

the newly allocated buffer state object.

### 7.9.3.3 YY\_BUFFER\_STATE DSExpressionFlex\_scan\_bytes (yyconst char \* *yybytes*, yy\_size\_t *\_yybytes\_len*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan the given bytes. The next call to DSExpressionFlexlex() will scan from a *copy* of *bytes*.

#### Parameters

*bytes* the byte buffer to scan  
*len* the number of bytes in the buffer pointed to by *bytes*.  
*yyscanner* The scanner object.

#### Returns

the newly allocated buffer state object.

### 7.9.3.4 YY\_BUFFER\_STATE DSExpressionFlex\_scan\_string (yyconst char \* *yyst*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan a string. The next call to DSExpressionFlexlex() will scan from a *copy* of *str*.

#### Parameters

*yyst* a NUL-terminated string to scan  
*yyscanner* The scanner object.

#### Returns

the newly allocated buffer state object.

#### Note

If you want to scan bytes that may contain NUL values, then use [DSExpressionFlex\\_scan\\_bytes\(\)](#) instead.

### 7.9.3.5 int DSExpressionFlexget\_column (yyscan\_t *yyscanner*)

Get the current column number.

#### Parameters

*yyscanner* The scanner object.

### 7.9.3.6 YY\_EXTRA\_TYPE DSExpressionFlexget\_extra (yyscan\_t *yyscanner*)

Get the user-defined data for this scanner.

#### Parameters

*yyscanner* The scanner object.

### 7.9.3.7 FILE \* DSExpressionFlexget\_in (yyscan\_t *yyscanner*)

Get the input stream.

#### Parameters

*yyscanner* The scanner object.

### 7.9.3.8 yy\_size\_t DSExpressionFlexget\_leng (yyscan\_t *yyscanner*)

Get the length of the current token.

#### Parameters

*yyscanner* The scanner object.

### 7.9.3.9 int DSExpressionFlexget\_lineno (yyscan\_t *yyscanner*)

Get the current line number.

#### Parameters

*yyscanner* The scanner object.

### 7.9.3.10 FILE \* DSExpressionFlexget\_out (yyscan\_t *yyscanner*)

Get the output stream.

#### Parameters

*yyscanner* The scanner object.

**7.9.3.11 char \* DSExpressionFlexget\_text (yyvsp\_t yyscanner)**

Get the current token.

**Parameters**

*yyscanner* The scanner object.

**7.9.3.12 void DSExpressionFlexpop\_buffer\_state (yyvsp\_t yyscanner)**

Removes and deletes the top of the stack, if present. The next element becomes the new top.

**Parameters**

*yyscanner* The scanner object.

**7.9.3.13 void DSExpressionFlexpush\_buffer\_state (YY\_BUFFER\_STATE *new\_buffer*, yyscan\_t yyscanner)**

Pushes the new state onto the stack. The new state becomes the current state. This function will allocate the stack if necessary.

**Parameters**

*new\_buffer* The new state.

*yyscanner* The scanner object.

**7.9.3.14 void DSExpressionFlexset\_column (int *column\_no*, yyscan\_t yyscanner)**

Set the current column.

**Parameters**

*line\_number*

*yyscanner* The scanner object.

**7.9.3.15 void DSExpressionFlexset\_extra (YY\_EXTRA\_TYPE *user\_defined*, yyscan\_t yyscanner)**

Set the user-defined data. This data is never touched by the scanner.

**Parameters**

*user\_defined* The data to be associated with this scanner.

*yyscanner* The scanner object.

**7.9.3.16 void DSExpressionFlexset\_in (FILE \* *in\_str*, yyscan\_t *yyscanner*)**

Set the input stream. This does not discard the current input buffer.

**Parameters**

*in\_str* A readable stream.

*yyscanner* The scanner object.

**See also**

DSExpressionFlex\_switch\_to\_buffer

**7.9.3.17 void DSExpressionFlexset\_lineno (int *line\_number*, yyscan\_t *yyscanner*)**

Set the current line number.

**Parameters**

*line\_number*

*yyscanner* The scanner object.

## 7.10 DSGMASystem.c File Reference

Implementation file with functions for dealing with GMA Systems.

```
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include "DSTypes.h"
#include "DSErrors.h"
#include "DSMemoryManager.h"
#include "DSGMASystem.h"
#include "DSExpression.h"
#include "DSExpressionTokenizer.h"
#include "DSGMASystemGrammar.h"
#include "DSMatrix.h"
#include "DSMatrixArray.h"
```

Include dependency graph for DSGMASystem.c: This graph shows which files directly or indirectly include this file:

### Defines

- #define **DS\_GMA\_EQUATION\_STR\_BUF** 1000
- #define **DSGMAXi**(x) ((x)->Xi)
- #define **DSGMAXd**(x) ((x)->Xd)
- #define **DSGMAAlpha**(x) ((x)->alpha)
- #define **DSGMABeta**(x) ((x)->beta)
- #define **DSGMAGd**(x) ((x)->Gd)
- #define **DSGMAGi**(x) ((x)->Gi)
- #define **DSGMAHd**(x) ((x)->Hd)
- #define **DSGMAHi**(x) ((x)->Hi)
- #define **DSGMASignature**(x) ((x)->signature)

### Functions

- **DSGMASystem** \* **DSGMASystemCopy** (const **DSGMASystem** \*gma)
- void **DSGMASystemFree** (**DSGMASystem** \*gma)
- **DSGMASystem** \* **DSGMASystemByParsingStringList** (const **DSVariablePool** \*const Xd, const char \*const string,...)
- **DSGMASystem** \* **DSGMASystemByParsingStrings** (const **DSVariablePool** \*const Xd, char \*const \*const strings, const DSUInteger numberOfEquations)
- **DSGMASystem** \* **DSGMASystemByParsingStringsWithXi** (const **DSVariablePool** \*const Xd, const **DSVariablePool** \*const Xi, char \*const \*const strings, const DSUInteger numberOfEquations)
- const DSUInteger **DSGMASystemNumberOfCases** (const **DSGMASystem** \*gma)
- const DSUInteger **DSGMASystemNumberOfEquations** (const **DSGMASystem** \*gma)
- **DSExpression** \*\* **DSGMASystemEquations** (const **DSGMASystem** \*gma)

- **DSExpression** \* **DSGMASystemPositiveTermsForEquations** (const **DSGMASystem** \*gma, const DSUInteger equation)
- **DSExpression** \* **DSGMASystemNegativeTermsForEquations** (const **DSGMASystem** \*gma, const DSUInteger equation)
- const **DSMatrix** \* **DSGMASystemAlpha** (const **DSGMASystem** \*gma)
- const **DSMatrix** \* **DSGMASystemBeta** (const **DSGMASystem** \*gma)
- const **DSMatrixArray** \* **DSGMASystemGd** (const **DSGMASystem** \*gma)
- const **DSMatrixArray** \* **DSGMASystemGi** (const **DSGMASystem** \*gma)
- const **DSMatrixArray** \* **DSGMASystemHd** (const **DSGMASystem** \*gma)
- const **DSMatrixArray** \* **DSGMASystemHi** (const **DSGMASystem** \*gma)
- const **DSVariablePool** \* **DSGMASystemXd** (const **DSGMASystem** \*gma)
- const **DSVariablePool** \* **DSGMASystemXi** (const **DSGMASystem** \*gma)
- const DSUInteger \* **DSGMASystemSignature** (const **DSGMASystem** \*gma)
- void **DSGMASystemPrint** (const **DSGMASystem** \*gma)
- void **DSGMASystemPrintEquations** (const **DSGMASystem** \*gma)

### 7.10.1 Detailed Description

Implementation file with functions for dealing with GMA Systems. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.11 DSGMASystem.h File Reference

Header file with functions for dealing with GMA Systems.

```
#include "DSTypes.h"
#include "DSVariable.h"
```

Include dependency graph for DSGMASystem.h: This graph shows which files directly or indirectly include this file:

### Defines

- `#define M_DS_GMA_NULL M_DS_NULL` ": GMA System is NULL"

### Functions

- `DSGMASystem * DSGMASystemCopy` (const `DSGMASystem` \*gma)
- `void DSGMASystemFree` (`DSGMASystem` \*gma)
- `DSGMASystem * DSGMASystemByParsingStringList` (const `DSVariablePool` \*const Xd, const char \*const string,...)
- `DSGMASystem * DSGMASystemByParsingStrings` (const `DSVariablePool` \*const Xd, char \*const \*const strings, const `DSUInteger` numberOfEquations)
- `DSGMASystem * DSGMASystemByParsingStringsWithXi` (const `DSVariablePool` \*const Xd, const `DSVariablePool` \*const Xi, char \*const \*const strings, const `DSUInteger` numberOfEquations)
- const `DSUInteger DSGMASystemNumberOfEquations` (const `DSGMASystem` \*gma)
- `DSExpression ** DSGMASystemEquations` (const `DSGMASystem` \*gma)
- `DSExpression * DSGMASystemPositiveTermsForEquations` (const `DSGMASystem` \*gma, const `DSUInteger` equation)
- `DSExpression * DSGMASystemNegativeTermsForEquations` (const `DSGMASystem` \*gma, const `DSUInteger` equation)
- const `DSMatrix * DSGMASystemAlpha` (const `DSGMASystem` \*gma)
- const `DSMatrix * DSGMASystemBeta` (const `DSGMASystem` \*gma)
- const `DSMatrixArray * DSGMASystemGd` (const `DSGMASystem` \*gma)
- const `DSMatrixArray * DSGMASystemGi` (const `DSGMASystem` \*gma)
- const `DSMatrixArray * DSGMASystemHd` (const `DSGMASystem` \*gma)
- const `DSMatrixArray * DSGMASystemHi` (const `DSGMASystem` \*gma)
- const `DSVariablePool * DSGMASystemXd` (const `DSGMASystem` \*gma)
- const `DSVariablePool * DSGMASystemXi` (const `DSGMASystem` \*gma)
- const `DSUInteger DSGMASystemNumberOfCases` (const `DSGMASystem` \*gma)
- const `DSUInteger * DSGMASystemSignature` (const `DSGMASystem` \*gma)
- `void DSGMASystemPrint` (const `DSGMASystem` \*gma)
- `void DSGMASystemPrintEquations` (const `DSGMASystem` \*gma)

### 7.11.1 Detailed Description

Header file with functions for dealing with GMA Systems. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).



The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.12 DSGMASystemParsingAux.h File Reference

Implementation file with functions for dealing with the parsing of GMA Systems.

```
#include "DSTypes.h"
```

Include dependency graph for DSGMASystemParsingAux.h:

### Data Structures

- struct [parser\\_aux](#)  
*Data type used to parse strings to GMA System.*
- union [base\\_info](#)

### Defines

- #define **AUX\_EXPONENT\_CONSTANT\_BASE** NAN
- #define **AUX\_SIGN\_UNDEFINED** '?'
- #define **AUX\_SIGN\_POSITIVE** '+'
- #define **AUX\_SIGN\_NEGATIVE** '-'
- #define **AUX\_PARSER\_FAILED** false
- #define **AUX\_PARSER\_SUCCESS** true
- #define **DSGMAParserAuxNumberOfBases**(x) (x->numberOfBases)
- #define **DSGMAParserAuxBaseAtIndexIsVariable**(x, y) (!isnan(DSGMAParserAuxExponentAtIndex(x, y)))
- #define **DSGMAParserAuxSetParserFailed**(x) ((x)->succeeded = false)

### Typedefs

- typedef struct [parser\\_aux](#) [gma\\_parseraux\\_t](#)  
*Data type used to parse strings to GMA System.*

### Functions

- [gma\\_parseraux\\_t](#) \* **DSGMAParserAuxAlloc** (void)
- void **DSGMAParserAuxFree** ([gma\\_parseraux\\_t](#) \*root)
- void **DSGMAParserAuxNewTerm** ([gma\\_parseraux\\_t](#) \*current)
- [gma\\_parseraux\\_t](#) \* **DSGMAParserAuxNextNode** (const [gma\\_parseraux\\_t](#) \*const aux)
- void **DSGMAParserAuxSetSign** ([gma\\_parseraux\\_t](#) \*aux, const char sign)
- void **DSGMAParserAuxAddVariableExponentPair** ([gma\\_parseraux\\_t](#) \*aux, const char \*const name, const double exponent)
- void **DSGMAParserAuxAddConstantBase** ([gma\\_parseraux\\_t](#) \*aux, const double base)
- const char **DSGMAParserAuxSign** (const [gma\\_parseraux\\_t](#) \*const aux)
- const double **DSGMAParserAuxExponentAtIndex** (const [gma\\_parseraux\\_t](#) \*const aux, const DSUInteger index)
- const char \*const **DSGMAParserAuxVariableAtIndex** (const [gma\\_parseraux\\_t](#) \*const aux, const DSUInteger index)

- const double **DSGMAParseAuxsConstantBaseAtIndex** (const [gma\\_parseraux\\_t](#) \*const aux, const DSUInteger index)
- const bool **DSGMAParserAuxParsingFailed** (const [gma\\_parseraux\\_t](#) \*const aux)

### 7.12.1 Detailed Description

Implementation file with functions for dealing with the parsing of GMA Systems. Header file with functions for dealing with the parsing of GMA Systems.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.12.2 Typedef Documentation

#### 7.12.2.1 typedef struct parser\_aux gma\_parseraux\_t

Data type used to parse strings to GMA System.

This data structure forms an organized list of terms, each with base exponent pairs that are then used to create the system matrices. This data structure is key for the parsing of GMA systems. Each node in the [gma\\_parseraux\\_t](#) list represent a term in an expression in the order it was found, and each node points to the next term. Each expression, or equation, has it's own list of terms. If a base is a constant, then it should not have an exponent, and hence it's exponent is assigned a NAN value and this is used to indicate that the base is a constant.

## 7.13 DSIO.c File Reference

Implementation file with standard input and output functions.

```
#include <stdio.h>
#include <string.h>
#include "DSIO.h"
#include "DSMemoryManager.h"
#include "DSVariable.h"
#include "DSMatrix.h"
#include "DSMatrixArray.h"
#include "DSGMASystem.h"
#include "DSSSystem.h"
#include "DSCase.h"
```

Include dependency graph for DSIO.c:

### Defines

- `#define DS_IO_TAG_TYPE_Matrix` `"\DSMatrix\"`
- `#define DS_IO_TAG_TYPE_MatrixArray` `"\DSMatrixArray\"`
- `#define DS_IO_TAG_TYPE_VariablePool` `"\DSVariablePool\"`
- `#define DS_IO_TAG_TYPE_Dictionary` `"\DSDictionary\"`
- `#define DS_IO_TAG_TYPE_SSystem` `"\DSSSystem\"`
- `#define DS_IO_TAG_TYPE_Case` `"\DSCase\"`
- `#define DS_IO_TAG_TYPE_DesignSpace` `"\DSDesignSpace\"`

### Functions

- void [DSIOSetErrorFile](#) (FILE \*aFile)  
*Function to assign default error file.*
- void [DSIOSetPrintFunction](#) (int(\*printFunction)(const char \*,...))  
*Function to assign default printf function.*
- void [DSIOSetPostWarningFunction](#) (void(\*warningFunction)(const char \*message))  
*Function to assign default warning posting function.*
- void [DSIOSetPostErrorFunction](#) (void(\*errorFunction)(const char \*message))  
*Function to assign default error posting function.*
- void [DSIOSetPostFatalErrorFunction](#) (void(\*fatalErrorFunction)(const char \*message))  
*Function to assign default fatal error posting function.*
- void [DSIOSetCaseJSONOptions](#) (const DSUInteger options)  
*Function that sets the conversion options for a [DSCase](#) to JSON format.*

- void [DSIOSetSSystemJSONOptions](#) (const DSUInteger options)  
*Function that sets the conversion options for a [DSSSystem](#) to JSON format.*
- char \* [DSVariablePoolStringInJSONFormat](#) (const DSVariablePool \*pool)  
*Function to convert a [DSVariablePool](#) into a JSON formatted string.*
- char \* [DSMatrixStringInJSONFormat](#) (const DSMatrix \*matrix)  
*Function to convert a [DSMatrix](#) into a JSON formatted string.*
- char \* [DSMatrixArrayStringInJSONFormat](#) (const DSMatrixArray \*array)  
*Function to convert a [DSMatrixArray](#) into a JSON formatted string.*
- char \* [DSSSystemStringInJSONFormat](#) (const DSSSystem \*ssys)  
*Function to convert a [DSSSystem](#) into a JSON formatted string.*
- char \* [DSCaseStringInJSONFormat](#) (const DSCase \*aCase)  
*Function to convert a [DSCase](#) into a JSON formatted string.*

## Variables

- DSUInteger [DSSSystemPrintingOptions](#)  
*Variable with flags controlling S-System to JSON string conversion.*
- DSUInteger [DSCasePrintingOptions](#)  
*Variable with flags controlling the conversion of a Case to a JSON string.*

### 7.13.1 Detailed Description

Implementation file with standard input and output functions. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.13.2 Function Documentation

### 7.13.2.1 `char* DSCaseStringInJSONFormat (const DSCase * aCase)`

Function to convert a [DSCase](#) into a JSON formatted string.

This function is used to convert a [DSCase](#) into a JSON object. The [DSCase](#) is represented with a set of objects, where each object is a field of the [DSCase](#) object. The default behavior exports all of the fields, this behavior can be overwritten by changing the [DSCase](#) conversion options.

#### Parameters

*aCase* A [DSCase](#) that will be used to create the JSON object.

#### Returns

A C string with the JSON formatted data. If NULL, the conversion failed.

#### See also

[DSIOSetCaseJSONOptions\(\)](#)

### 7.13.2.2 `void DSIOSetCaseJSONOptions (const DSUInteger options)`

Function that sets the conversion options for a [DSCase](#) to JSON format.

This function is used to overwrite the default export behavior of the [DSCase](#) object. The default behavior converts all of the data fields of the [DSCase](#) into a JSON format, these options can be changed so the JSON conversion only includes some fields, such as excluding the conditions, excluding the S-System, etc.

#### Parameters

*options* A DSUInteger with the option flags, as specified by the [DSCase](#) options.

#### See also

[Options for JSON conversion of DSCase object.](#)

### 7.13.2.3 `void DSIOSetErrorFile (FILE * aFile)`

Function to assign default error file.

This function is used to assign the default error file, DSIOErrorFile. Changing the error file should be done via this function, as it circumvents potential problems associated with dynamic linking.

#### Parameters

*aFile* A FILE \* that will be used to write error messages when the default error posting mechanism is used.

#### See also

[DSIOSetPostWarningFunction](#)  
[DSIOSetPostErrorFunction](#)  
[DSIOSetPostFatalErrorFunction](#)  
[DSError](#)

**7.13.2.4 void DSIOSetPostErrorFunction (void(\*) (const char \*message) errorFunction)**

Function to assign default error posting function.

This function is used to assign the function that handles the errors generated from the design space toolbox. Internally, it assigns the global variable DSPostError which points to a function.

**Parameters**

**errorFunction** A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

**7.13.2.5 void DSIOSetPostFatalErrorFunction (void(\*) (const char \*message) fatalErrorFunction)**

Function to assign default fatal error posting function.

This function is used to assign the function that handles the fatal errors generated from the design space toolbox. Internally, it assigns the global variable DSPostFatalError which points to a function.

**Parameters**

**errorFunction** A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

**7.13.2.6 void DSIOSetPostWarningFunction (void(\*) (const char \*message) warningFunction)**

Function to assign default warning posting function.

This function is used to assign the function that handles the warnings generated from the design space toolbox. Internally, it assigns the global variable DSPostWarning which points to a function.

**Parameters**

**warningFunction** A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

**7.13.2.7 void DSIOSetPrintFunction (int(\*) (const char \*,...) printfFunction)**

Function to assign default printf function.

This function is used to assign the formatted print function, DSprintf. This function assigns the DSprintf pointer to the function that should be used to print formatted strings. This function MUST be used to avoid problems relating to dynamic linking; by using this function the global variable DSprintf is loaded into memory prior to changing its value.

**Parameters**

**printfFunction** A pointer to a function of the form int function(const char \*, ...). If NULL, default behavior is restored.

### 7.13.2.8 void DSIOSetSSystemJSONOptions (const DSUInteger *options*)

Function that sets the conversion options for a [DSSSystem](#) to JSON format.

This function is used to overwrite the default export behavior of the [DSSSystem](#) object. The default behavior converts all of the data fields of the S-System into a JSON format, these options can be changed so the JSON conversion only includes some fields, such as excluding the solution.

#### Parameters

*options* A DSUInteger with the option flags, as specified by the [DSSSystem](#) options.

#### See also

[Options for JSON conversion of DSSSystem object.](#)

### 7.13.2.9 char\* DSMatrixArrayStringInJSONFormat (const DSMatrixArray \* *array*)

Function to convert a [DSMatrixArray](#) into a JSON formatted string.

This function is used to convert a [DSMatrix](#) into a JSON object. The matrix array is stored as an array of objects, where each object is a [DSMatrix](#). The order of the [DSMatrix](#) object in the array represent the order of matrices in the matrix array.

#### Parameters

*array* A [DSMatrixArray](#) that will be used to create the JSON object.

#### Returns

A C string with the JSON formatted data. If NULL, the conversion failed.

### 7.13.2.10 char\* DSMatrixStringInJSONFormat (const DSMatrix \* *matrix*)

Function to convert a [DSMatrix](#) into a JSON formatted string.

This function is used to convert a [DSMatrix](#) into a JSON object. The matrix is stored as an array of arrays. The array of arrays represents the rows of the matrix, whereas the arrays of value are the values at the columns for a particular row.

#### Parameters

*matrix* A [DSMatrix](#) that will be used to create the JSON object.

#### Returns

A C string with the JSON formatted data. If NULL, the conversion failed.

### 7.13.2.11 char\* DSSSystemStringInJSONFormat (const DSSSystem \* *ssys*)

Function to convert a [DSSSystem](#) into a JSON formatted string.

This function is used to convert a [DSSSystem](#) into a JSON object. The S-System as a set of objects, where each object represents each of the fields of the [DSSSystem](#). The default behavior exports all of the fields, this behavior can be overwritten by changing the S-System conversion options.



**Parameters**

*ssys* A [DSSSystem](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

**See also**

[DSIOSetSSystemJSONOptions\(\)](#)

**7.13.2.12 char\* DSVariablePoolStringInJSONFormat (const DSVariablePool \* pool)**

Function to convert a [DSVariablePool](#) into a JSON formatted string.

This function is used to convert a [DSVariablePool](#) into a JSON object. The variables of the variable pool are stored as pairs of a string and value.

**Parameters**

*pool* A [DSVariablePool](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

**7.13.3 Variable Documentation****7.13.3.1 DSUInteger DSCasePrintingOptions**

Variable with flags controlling the conversion of a Case to a JSON string.

This global variable is checked when converting a Case structure to a JSON string. This variable will check several flags as specified by DS\_CASE\_JSON\_OPTIONS. The default value of the variable indicates that all the properties will be included in the JSON string.

**See also**

[Options for JSON conversion of DSCase object.](#)  
[DSIOSetCaseJSONOptions\(\)](#)

**7.13.3.2 DSUInteger DSSSystemPrintingOptions**

Variable with flags controlling S-System to JSON string conversion.

This global variable is checked when converting a S-System structure to a JSON string. This variable will check several flags as specified by DS\_SSYSTEM\_JSON\_OPTIONS. The default value of the variable indicates that all the properties will be included in the JSON string.

**See also**

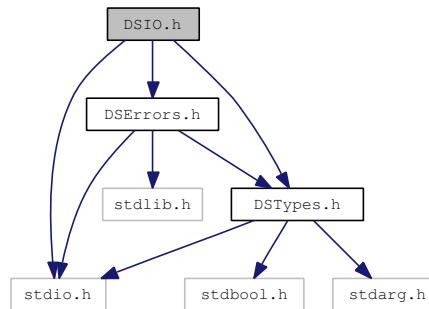
[Options for JSON conversion of DSSSystem object.](#)  
[DSIOSetSSystemJSONOptions\(\)](#)

## 7.14 DSIO.h File Reference

Header file with standard input and output functions.

```
#include <stdio.h>
#include "DSTypes.h"
```

Include dependency graph for DSIO.h:



This graph shows which files directly or indirectly include this file:

### Defines

- `#define DS_CASE_JSON_NO_SSYSTEM 1`  
*Flag value indicating that the S-System information should not be included in the JSON string.*
- `#define DS_CASE_JSON_NO_CASE_SIGNATURE 2`  
*Flag value indicating that the case signature should not be included in the JSON string.*
- `#define DS_CASE_JSON_NO_CONDITIONS 4`  
*Flag value indicating that the conditions for validity should not be included in the JSON string.*
- `#define DS_SSYSTEM_JSON_NO_SOLUTION 1`  
*Flag value indicating that the S-System solution should not be included in the JSON string.*
- `#define DS_SSYSTEM_JSON_NO_SINGULAR 2`  
*Flag value indicating that the JSON string will not indicate if the S-System is singular.*

### Functions

- `void DSIOSetErrorFile (FILE *aFile)`  
*Function to assign default error file.*
- `void DSIOSetPrintFunction (int(*printFunction)(const char *,...))`  
*Function to assign default printf function.*
- `void DSIOSetPostWarningFunction (void(*warningFunction)(const char *message))`  
*Function to assign default warning posting function.*

- void [DSIOSetPostErrorFunction](#) (void(\*errorFunction)(const char \*message))  
*Function to assign default error posting function.*
- void [DSIOSetPostFatalErrorFunction](#) (void(\*fatalErrorFunction)(const char \*message))  
*Function to assign default fatal error posting function.*
- void [DSIOSetCaseJSONOptions](#) (const DSUInteger options)  
*Function that sets the conversion options for a [DSCase](#) to JSON format.*
- void [DSIOSetSSystemJSONOptions](#) (const DSUInteger options)  
*Function that sets the conversion options for a [DSSSystem](#) to JSON format.*
- char \* [DSVariablePoolStringInJSONFormat](#) (const [DSVariablePool](#) \*pool)  
*Function to convert a [DSVariablePool](#) into a JSON formatted string.*
- char \* [DSMatrixStringInJSONFormat](#) (const [DSMatrix](#) \*matrix)  
*Function to convert a [DSMatrix](#) into a JSON formatted string.*
- char \* [DSMatrixArrayStringInJSONFormat](#) (const [DSMatrixArray](#) \*array)  
*Function to convert a [DSMatrixArray](#) into a JSON formatted string.*
- char \* [DSSSystemStringInJSONFormat](#) (const [DSSSystem](#) \*ssys)  
*Function to convert a [DSSSystem](#) into a JSON formatted string.*
- char \* [DSCaseStringInJSONFormat](#) (const [DSCase](#) \*aCase)  
*Function to convert a [DSCase](#) into a JSON formatted string.*
- [DSVariablePool](#) \* [DSVariablePoolByParsingStringInJSONFormat](#) (const char \*string)
- [DSMatrix](#) \* [DSMatrixByParsingStringInJSONFormat](#) (const char \*string)
- [DSMatrixArray](#) \* [DSMatrixArrayByParsingStringInJSONFormat](#) (const char \*string)
- [DSSSystem](#) \* [DSSSystemByParsingStringInJSONFormat](#) (const char \*string)
- [DSCase](#) \* [DSCaseByParsingStringInJSONFormat](#) (const char \*string)

## Variables

- int(\* [DSPrintf](#) )(const char \*,...)  
*Pointer to a function determining how messages are printed.*
- void(\* [DSPostWarning](#) )(const char \*message)  
*Pointer to a function determining how warning are handled.*
- void(\* [DSPostError](#) )(const char \*message)  
*Pointer to a function determining how errors are handled.*
- void(\* [DSPostFatalError](#) )(const char \*message)  
*Pointer to a function determining how fatal errors are handled.*
- FILE \* [DSIOErrorFile](#)  
*FILE pointer used for default error/warning printing.*

### 7.14.1 Detailed Description

Header file with standard input and output functions. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

#### Todo

- Define standard input and output file formats.
- Define criteria for warnings, errors and fatal errors.

### 7.14.2 Function Documentation

#### 7.14.2.1 `char* DSCaseStringInJSONFormat (const DSCase * aCase)`

Function to convert a [DSCase](#) into a JSON formatted string.

This function is used to convert a [DSCase](#) into a JSON object. The [DSCase](#) is represented with a set of objects, where each object is a field of the [DSCase](#) object. The default behavior exports all of the fields, this behavior can be overwritten by changing the [DSCase](#) conversion options.

#### Parameters

*aCase* A [DSCase](#) that will be used to create the JSON object.

#### Returns

A C string with the JSON formatted data. If NULL, the conversion failed.

#### See also

[DSIOSetCaseJSONOptions\(\)](#)

#### 7.14.2.2 `void DSIOSetCaseJSONOptions (const DSUInteger options)`

Function that sets the conversion options for a [DSCase](#) to JSON format.

This function is used to overwrite the default export behavior of the [DSCase](#) object. The default behavior converts all of the data fields of the [DSCase](#) into a JSON format, these options can be changed so the JSON conversion only includes some fields, such as excluding the conditions, excluding the S-System, etc.

**Parameters**

*options* A DSUInteger with the option flags, as specified by the [DSCase](#) options.

**See also**

[Options for JSON conversion of DSCase object.](#)

**7.14.2.3 void DSIOSetErrorFile (FILE \* *aFile*)**

Function to assign default error file.

This function is used to assign the default error file, DSIOErrorFile. Changing the error file should be done via this function, as it circumvents potential problems associated with dynamic linking.

**Parameters**

*aFile* A FILE \* that will be used to write error messages when the default error posting mechanism is used.

**See also**

[DSIOSetPostWarningFunction](#)  
[DSIOSetPostErrorFunction](#)  
[DSIOSetPostFatalErrorFunction](#)  
[DSError](#)

**7.14.2.4 void DSIOSetPostErrorFunction (void(\*)(const char \*message) *errorFunction*)**

Function to assign default error posting function.

This function is used to assign the function that handles the errors generated from the design space toolbox. Internally, it assigns the global variable DSPostError which points to a function.

**Parameters**

*errorFunction* A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

**7.14.2.5 void DSIOSetPostFatalErrorFunction (void(\*)(const char \*message) *fatalErrorFunction*)**

Function to assign default fatal error posting function.

This function is used to assign the function that handles the fatal errors generated from the design space toolbox. Internally, it assigns the global variable DSPostFatalError which points to a function.

**Parameters**

*errorFunction* A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

#### 7.14.2.6 void DSIOSetPostWarningFunction (void(\*) (const char \*message) *warningFunction*)

Function to assign default warning posting function.

This function is used to assign the function that handles the warnings generated from the design space toolbox. Internally, it assigns the global variable DSPostWarning which points to a function.

##### Parameters

***warningFunction*** A pointer to a function of the form void function(const char \*). If NULL, default behavior is restored.

#### 7.14.2.7 void DSIOSetPrintfFunction (int(\*) (const char \*,...) *printFunction*)

Function to assign default printf function.

This function is used to assign the formatted print function, DSPrintf. This function assigns the DSPrintf pointer to the function that should be used to print formatted strings. This function MUST be used to avoid problems relating to dynamic linking; by using this function the global variable DSPrintf is loaded into memory prior to changing its value.

##### Parameters

***printFunction*** A pointer to a function of the form int function(const char \*, ...). If NULL, default behavior is restored.

#### 7.14.2.8 void DSIOSetSSystemJSONOptions (const DSUInteger *options*)

Function that sets the conversion options for a [DSSSystem](#) to JSON format.

This function is used to overwrite the default export behavior of the [DSSSystem](#) object. The default behavior converts all of the data fields of the S-System into a JSON format, these options can be changed so the JSON conversion only includes some fields, such as excluding the solution.

##### Parameters

***options*** A DSUInteger with the option flags, as specified by the [DSSSystem](#) options.

##### See also

[Options for JSON conversion of DSSSystem object.](#)

#### 7.14.2.9 char\* DSMatrixArrayStringInJSONFormat (const DSMatrixArray \* *array*)

Function to convert a [DSMatrixArray](#) into a JSON formatted string.

This function is used to convert a [DSMatrix](#) into a JSON object. The matrix array is stored as an array of objects, where each object is a [DSMatrix](#). The order of the [DSMatrix](#) object in the array represent the order of matrices in the matrix array.

##### Parameters

***array*** A [DSMatrixArray](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

**7.14.2.10 char\* DSMatrixStringInJSONFormat (const DSMatrix \* *matrix*)**

Function to convert a [DSMatrix](#) into a JSON formatted string.

This function is used to convert a [DSMatrix](#) into a JSON object. The matrix is stored as an array of arrays. The array of arrays represents the rows of the matrix, whereas the arrays of value are the values at the columns for a particular row.

**Parameters**

*matrix* A [DSMatrix](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

**7.14.2.11 char\* DSSSystemStringInJSONFormat (const DSSSystem \* *ssys*)**

Function to convert a [DSSSystem](#) into a JSON formatted string.

This function is used to convert a [DSSSystem](#) into a JSON object. The S-System as a set of objects, where each object represents each of the fields of the [DSSSystem](#). The default behavior exports all of the fields, this behavior can be overwritten by changing the S-System conversion options.

**Parameters**

*ssys* A [DSSSystem](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

**See also**

[DSIOSetSSystemJSONOptions\(\)](#)

**7.14.2.12 char\* DSVariablePoolStringInJSONFormat (const DSVariablePool \* *pool*)**

Function to convert a [DSVariablePool](#) into a JSON formatted string.

This function is used to convert a [DSVariablePool](#) into a JSON object. The variables of the variable pool are stored as pairs of a string and value.

**Parameters**

*pool* A [DSVariablePool](#) that will be used to create the JSON object.

**Returns**

A C string with the JSON formatted data. If NULL, the conversion failed.

### 7.14.3 Variable Documentation

#### 7.14.3.1 FILE\* DSIOErrorFile

FILE pointer used for default error/warning printing.

This pointer to a FILE tells the error handling system which FILE to print the error messages to. If this pointer is NULL, then the system sets it to the stderr file. This variable is only used internally with the default behavior of DSErrorFunction. To change the error file, the function DSIOSetErrorFile should be used in order to avoid errors caused by dynamic linking. These errors involve changing the value of a global variable that has not yet been loaded by the linker.

See also

[DSIOSetErrorFile](#)  
[DSErrorFunction](#)

#### 7.14.3.2 void(\* DSPostError)(const char \*message)

Pointer to a function determining how errors are handled.

This pointer to a function is used by DSErrorFunction to post errors. This pointer should be used to allow better integration of errors in programs that make use of the DesignSpaceToolbox. The function takes one argument, a constant C string with the error message. To change the function used, the function DSIOSetPostErrorFunction should be used. This is to avoid errors caused by dynamic linking. These errors involve changing the value of a global variable that has not yet been loaded by the linker.

See also

[DSIOSetPostErrorFunction](#)

#### 7.14.3.3 void(\* DSPostFatalError)(const char \*message)

Pointer to a function determining how fatal errors are handled.

This pointer to a function is used by DSErrorFunction to post fatal errors. This pointer should be used to allow better integration of errors in programs that make use of the DesignSpaceToolbox. The function takes one argument, a constant C string with the error message. To change the function used, the function DSIOSetPostFatalErrorFunction should be used. This is to avoid errors caused by dynamic linking. These errors involve changing the value of a global variable that has not yet been loaded by the linker.

See also

[DSIOSetPostErrorFunction](#)

#### 7.14.3.4 void(\* DSPostWarning)(const char \*message)

Pointer to a function determining how warnings are handled.

This pointer to a function is used by DSErrorFunction to post warnings. This pointer should be used to allow better integration of warnings in programs that make use of the DesignSpaceToolbox. The function takes one argument, a constant C string with the warning message. To change the function used, the function DSIOSetPostWarningFunction should be used. This is to avoid errors caused by dynamic linking. These errors involve changing the value of a global variable that has not yet been loaded by the linker.



**See also**[DSIOSetPostWarningFunction](#)**7.14.3.5 int(\* DSPrintf)(const char \*,...)**

Pointer to a function determining how messages are printed.

This pointer to a function tells the error handling system which function to call with the error messages. If this pointer is NULL, the design space toolbox should have a default printing format, using printf to stdout. This pointer is intended to be used to override default behavior to be override. An example could be by using the mexPrintf function in matlab.

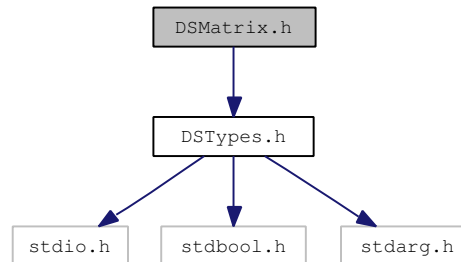
**See also**[DSIOSetPrintFunction](#)

## 7.15 DSMatrix.h File Reference

Header file with functions for dealing with matrices.

```
#include "DSTypes.h"
```

Include dependency graph for DSMatrix.h:



This graph shows which files directly or indirectly include this file:

### Defines

- `#define M_DS_MAT_NULL` "Pointer to matrix is NULL"  
*Message for a NULL [DSMatrix](#) pointer.*
- `#define M_DS_MAT_OUTOFBOUNDS` "Row or column out of bounds"  
*Message for a row or column exceeding matrix bounds.*
- `#define M_DS_MAT_NOINTERNAL` "Matrix data is empty"  
*Message for a NULL internal matrix structure.*
- `#define DSMatrixRows(x) ((x)->rows)`
- `#define DSMatrixColumns(x) ((x)->columns)`
- `#define DSMatrixInternalPointer(x) ((x)->mat)`

### Enumerations

- `enum { __MAT_GSL__, __MAT_CLAPACK__ }`

### Functions

- `DSMatrix * DSMatrixAlloc` (const DSUInteger rows, const DSUInteger columns)  
*Memory allocation for a [DSMatrix](#) using malloc.*
- `DSMatrix * DSMatrixCalloc` (const DSUInteger rows, const DSUInteger columns)  
*Memory allocation for a [DSMatrix](#) using calloc.*
- `DSMatrix * DSMatrixCopy` (const [DSMatrix](#) \*original)  
*Copies a [DSMatrix](#).*

- void **DSMatrixFree** (**DSMatrix** \*matrix)  
*Freeing memory for **DSMatrix**.*
- **DSMatrix** \* **DSMatrixIdentity** (const DSUInteger size)  
*Allocates a new **DSMatrix** as an identity matrix.*
- **DSMatrix** \* **DSMatrixRandomNumbers** (const DSUInteger rows, const DSUInteger columns)  
*Allocates a new **DSMatrix** with random values between 0 and 1.*
- **DSMatrix** \* **DSMatrixByParsingString** (const char \*string)  
*Creates a new matrix by parsing a tab-delimited matrix.*
- **DSMatrix** \* **DSMatrixBySubtractingMatrix** (const **DSMatrix** \*lvalue, const **DSMatrix** \*rvalue)  
*Create a new **DSMatrix** object by subtracting a matrix from another.*
- **DSMatrix** \* **DSMatrixByAddingMatrix** (const **DSMatrix** \*lvalue, const **DSMatrix** \*rvalue)  
*Create a new **DSMatrix** object by adding a matrix to another.*
- **DSMatrix** \* **DSMatrixByDividingMatrix** (const **DSMatrix** \*lvalue, const **DSMatrix** \*rvalue)
- **DSMatrix** \* **DSMatrixByMultiplyingMatrix** (const **DSMatrix** \*lvalue, const **DSMatrix** \*rvalue)
- **DSMatrix** \* **DSMatrixByApplyingFunction** (const **DSMatrix** \*mvalue, double(\*function)(double))
- **DSMatrix** \* **DSMatrixBySubtractingScalar** (const **DSMatrix** \*lvalue, const double rvalue)
- **DSMatrix** \* **DSMatrixByAddingScalar** (const **DSMatrix** \*lvalue, const double rvalue)
- **DSMatrix** \* **DSMatrixByDividingScalar** (const **DSMatrix** \*lvalue, const double rvalue)
- **DSMatrix** \* **DSMatrixByMultiplyingScalar** (const **DSMatrix** \*lvalue, const double rvalue)
- double **DSMatrixDoubleValue** (const **DSMatrix** \*matrix, const DSUInteger row, const DSUInteger column)  
*Returns the element of the **DSMatrix** specified by a row and column.*
- void **DSMatrixSetDoubleValue** (**DSMatrix** \*matrix, const DSUInteger row, const DSUInteger column, const double value)
- void **DSMatrixSetDoubleValueAll** (**DSMatrix** \*matrix, const double value)  
*Sets all the values of a matrix to a value.*
- void **DSMatrixSetDoubleValuesList** (**DSMatrix** \*matrix, bool byColumns, DSUInteger numberOfValues, double firstValue,...)
- void **DSMatrixSetDoubleValues** (**DSMatrix** \*matrix, bool byColumns, DSUInteger numberOfValues, double \*values)
- void **DSMatrixRoundToSignificantFigures** (**DSMatrix** \*matrix, const unsigned char figures)
- **DSMatrix** \* **DSMatrixSubMatrixExcludingColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfColumns, const DSUInteger firstColumn,...)
- **DSMatrix** \* **DSMatrixSubMatrixExcludingColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfColumns, const DSUInteger \*columns)
- **DSMatrix** \* **DSMatrixSubMatrixExcludingRowList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger firstRow,...)
- **DSMatrix** \* **DSMatrixSubMatrixExcludingRows** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger \*rows)
- **DSMatrix** \* **DSMatrixSubMatrixIncludingRowList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger firstRow,...)

- **DSMatrix \* DSMatrixSubMatrixIncludingRows** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger \*rows)
- **DSMatrix \* DSMatrixSubMatrixIncludingColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfColumns, const DSUInteger firstColumn,...)
- **DSMatrix \* DSMatrixSubMatrixExcludingRowAndColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfColumns, const DSUInteger firstRow,...)
- **DSMatrix \* DSMatrixSubMatrixExcludingRowsAndColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfColumns, const DSUInteger \*rows, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixSubMatrixIncludingColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfColumns, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixSubMatrixIncludingRowAndColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfColumns, const DSUInteger firstRow,...)
- **DSMatrix \* DSMatrixAppendMatrices** (const **DSMatrix** \*firstMatrix, const **DSMatrix** \*secondMatrix, const bool byColumn)
- void **DSMatrixSwitchRows** (**DSMatrix** \*matrix, const DSUInteger rowA, const DSUInteger rowB)
- void **DSMatrixSwitchColumns** (**DSMatrix** \*matrix, const DSUInteger columnA, const DSUInteger columnB)
- **DSMatrix \* DSMatrixWithUniqueRows** (const **DSMatrix** \*matrix)
- void **DSMatrixPrint** (const **DSMatrix** \*matrix)
- bool **DSMatrixIsIdentity** (const **DSMatrix** \*matrix)
- bool **DSMatrixIsSquare** (const **DSMatrix** \*matrix)
- DSUInteger **DSMatrixRank** (const **DSMatrix** \*matrix)
- double **minimumValue** (const **DSMatrix** \*matrix, const bool shouldExcludeZero)
- double **maximumValue** (const **DSMatrix** \*matrix, const bool shouldExcludeZero)
- void **DSMatrixSubtractByMatrix** (**DSMatrix** \*addTo, const **DSMatrix** \*addBy)
- void **DSMatrixAddByMatrix** (**DSMatrix** \*addTo, const **DSMatrix** \*addBy)
- void **DSMatrixApplyFunction** (**DSMatrix** \*matrix, double(\*function)(double))
- void **DSMatrixMultiplyByScalar** (**DSMatrix** \*matrix, const double value)
- double **DSMatrixDeterminant** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixTranspose** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixInverse** (const **DSMatrix** \*matrix)
- **DSMatrixArray \* DSMatrixSVD** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixRightNullspace** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixLeftNullspace** (const **DSMatrix** \*matrix)
- **DSMatrixArray \* DSMatrixPLUDecomposition** (const **DSMatrix** \*matrix)  
*Creates a LU decomposition and returns the permutation matrix.*
- double \* **DSMatrixDataForGLPK** (const **DSMatrix** \*matrix)
- int \* **DSMatrixRowsForGLPK** (const **DSMatrix** \*matrix)
- int \* **DSMatrixColumnsForGLPK** (const **DSMatrix** \*matrix)

### 7.15.1 Detailed Description

Header file with functions for dealing with matrices. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.15.2 Function Documentation

### 7.15.2.1 DSMatrix\* DSMatrixAlloc (const DSUInteger *rows*, const DSUInteger *columns*)

Memory allocation for a [DSMatrix](#) using malloc.

Creates a new matrix of a particular size. The matrix that is allocated has all the values of the matrix defaulted to 0. The internal matrix pointer must be set to NULL; otherwise, the size of the matrix cannot be changed.

**Parameters**

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

**Returns**

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

### 7.15.2.2 DSMatrix\* DSMatrixByAddingMatrix (const DSMatrix \* *lvalue*, const DSMatrix \* *rvalue*)

Create a new [DSMatrix](#) object by adding a matrix to another.

This function takes two matrices of the same dimensions, and adds the ij element of the rvalue matrix to the ij element of the lvalue matrix. This function assumes constant matrices, and thus does not modify either of the inputs, but instead creates a copy of the first operand matrix, and calls DSMatrixAddByMatrix(), using the copy as the first operand.

**Parameters**

*lvalue* The first [DSMatrix](#) object to be added.

*rvalue* The second [DSMatrix](#) object to be added.

**Returns**

If the addition operation was succesful, the function returns a pointer to the newly allocated matrix. Otherwise, NULL is returned.

**See also**

DSMatrixAddByMatrix()

### 7.15.2.3 DSMatrix\* DSMatrixByParsingString (const char \* *string*)

Creates a new matrix by parsing a tab-delimited matrix.

This function reads an input string, containing rows delimited by tabs and columns delimited by newlines. This function generates a token stream, and thus checks the dimensions of the matrix prior to creating it.

#### Parameters

*string* A string containing the data to parse.

#### Returns

A [DSMatrix](#) data object with the parsed data. If parsing failed, returns NULL.

### 7.15.2.4 DSMatrix\* DSMatrixBySubtractingMatrix (const DSMatrix \* *lvalue*, const DSMatrix \* *rvalue*)

Create a new [DSMatrix](#) object by subtracting a matrix from another.

This function takes two matrices of the same dimensions, and subtracts the *ij* element of the *rvalue* matrix to the *ij* element of the *lvalue* matrix. This function assumes constant matrices, and thus does not modify either of the inputs, but instead creates a copy of the minuend operand matrix, and called `DSMatrixSubtractByMatrix()` with the copy as the new minuend.

#### Parameters

*lvalue* The [DSMatrix](#) object that is the minuend.

*rvalue* The [DSMatrix](#) object that is the subtrahend.

#### Returns

If the subtraction operation was successful, the function returns a pointer to the newly allocated difference matrix. Otherwise, NULL is returned.

#### See also

`DSMatrixSubtractByMatrix()`

### 7.15.2.5 DSMatrix\* DSMatrixCalloc (const DSUInteger *rows*, const DSUInteger *columns*)

Memory allocation for a [DSMatrix](#) using `calloc`.

Creates a new matrix of a particular size. The matrix that is allocated has all the values of the matrix defaulted to 0. The internal matrix pointer must be set to NULL; otherwise, the size of the matrix cannot be changed.

#### Parameters

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

#### Returns

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

### 7.15.2.6 DSMatrix\* DSMatrixCopy (const DSMatrix \* *original*)

Copies a [DSMatrix](#).

Creates a new matrix with the exact same size and contents as some other matrix. The new matrix is allocated, and thus must be freed.

#### Parameters

*original* The [DSMatrix](#) to be copied.

#### Returns

If the copy was succesful, a pointer to a copy of the [DSMatrix](#) is returned. Otherwise, NULL is returned.

### 7.15.2.7 double DSMatrixDoubleValue (const DSMatrix \* *matrix*, const DSUInteger *row*, const DSUInteger *column*)

Returns the element of the [DSMatrix](#) specified by a row and column.

Returns an element of the matrix, with indices i and j starting at 0.

#### Parameters

*matrix* The [DSMatrix](#) whose elements will be accessed.

*row* A DSUInteger specifying the row coordinate of the element to be accessed.

*column* A DSUInteger specifying the column coordinate of the element to be accessed.

#### Returns

If the value was succesfully retrieved, the double value contained at the row and column coordinate of the [DSMatrix](#) is returned. Otherwise, NaN is returned.

### 7.15.2.8 void DSMatrixFree (DSMatrix \* *matrix*)

Freeing memory for [DSMatrix](#).

Frees the memory associated with a [DSMatrix](#) data type. This function is a wrapper for the necessary steps needed to free the internal structure of the [DSMatrix](#) data type.

#### Parameters

*matrix* The [DSMatrix](#) to be freed.

### 7.15.2.9 DSMatrix\* DSMatrixIdentity (const DSUInteger *size*)

Allocates a new [DSMatrix](#) as an identity matrix.

Allocates a square matrix of a specified size, and initializes the diagonal values to 1 and all the other values to 0, creating an identity matrix. The new matrix is therefore an identity matrix.

**Parameters**

*size* A DSUInteger containing the number of rows and columns in the matrix.

**Returns**

If the identity matrix was successfully created, a pointer to the [DSMatrix](#) is returned. Otherwise, NULL is returned.

**7.15.2.10 DSMatrixArray\* DSMatrixPLUdecomposition (const DSMatrix \* A)**

Creates a LU decomposition and returns the permutation matrix.

This function creates a LU decomposition of a [DSMatrix](#) A. This function creates an array of three matrices: a [DSMatrix](#) P, a [DSMatrix](#) L and a [DSMatrix](#) U; where  $PA = LU$ .

**Parameters**

*A* A [DSMatrix](#) containing the matrix to be decomposed.

**7.15.2.11 DSMatrix\* DSMatrixRandomNumbers (const DSUInteger rows, const DSUInteger columns)**

Allocates a new [DSMatrix](#) with random values between 0 and 1.

Allocates a new [DSMatrix](#) with a specified size. The values of each of the entries in the matrix are randomly selected between 0 and 1.

**Parameters**

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

**Returns**

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

**7.15.2.12 void DSMatrixSetDoubleValueAll (DSMatrix \* matrix, const double value)**

Sets all the values of a matrix to a value.

This function does not allocate the necessary memory; instead it goes through all the rows and columns of the matrix, assigning them the specified value.

**Parameters**

*matrix* The [DSMatrix](#) that will be assigned the value.

*value* The double variable whose value will be assigned.

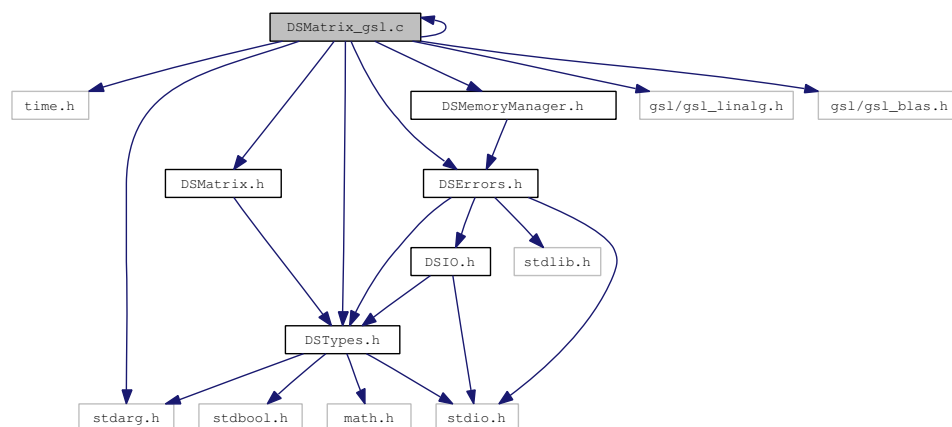


## 7.16 DSMatrix\_gsl.c File Reference

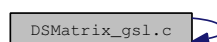
Implementation file with functions for dealing with matrices using the GNU Scientific Library (gsl).

```
#include <time.h>
#include <stdarg.h>
#include <string.h>
#include <unistd.h>
#include <gsl/gsl_linalg.h>
#include <gsl/gsl_blas.h>
#include "DSMatrix.h"
#include "DSErrors.h"
#include "DSMemoryManager.h"
#include "DSMatrixArray.h"
#include "DSMatrixTokenizer.h"
#include "DSTypes.h"
```

Include dependency graph for DSMatrix\_gsl.c:



This graph shows which files directly or indirectly include this file:



### Defines

- `#define DSMatrixSetRows(x, y) ((x)->rows = (y))`
- `#define DSMatrixSetColumns(x, y) ((x)->columns = (y))`

### Functions

- `DSMatrix * DSMatrixAlloc (const DSUInteger rows, const DSUInteger columns)`

*Memory allocation for a [DSMatrix](#) using malloc.*

- [DSMatrix](#) \* [DSMatrixCalloc](#) (const DSUInteger rows, const DSUInteger columns)  
*Memory allocation for a [DSMatrix](#) using calloc.*
- [DSMatrix](#) \* [DSMatrixCopy](#) (const [DSMatrix](#) \*original)  
*Copies a [DSMatrix](#).*
- void [DSMatrixFree](#) ([DSMatrix](#) \*matrix)  
*Freeing memory for [DSMatrix](#).*
- [DSMatrix](#) \* [DSMatrixIdentity](#) (const DSUInteger size)  
*Allocates a new [DSMatrix](#) as an identity matrix.*
- [DSMatrix](#) \* [DSMatrixRandomNumbers](#) (const DSUInteger rows, const DSUInteger columns)  
*Allocates a new [DSMatrix](#) with random values between 0 and 1.*
- [DSMatrix](#) \* [DSMatrixByParsingString](#) (const char \*string)  
*Creates a new matrix by parsing a tab-delimited matrix.*
- [DSMatrix](#) \* [DSMatrixBySubtractingMatrix](#) (const [DSMatrix](#) \*lvalue, const [DSMatrix](#) \*rvalue)  
*Create a new [DSMatrix](#) object by subtracting a matrix from another.*
- [DSMatrix](#) \* [DSMatrixByAddingMatrix](#) (const [DSMatrix](#) \*lvalue, const [DSMatrix](#) \*rvalue)  
*Create a new [DSMatrix](#) object by adding a matrix to another.*
- [DSMatrix](#) \* [DSMatrixByDividingMatrix](#) (const [DSMatrix](#) \*lvalue, const [DSMatrix](#) \*rvalue)
- [DSMatrix](#) \* [DSMatrixByMultiplyingMatrix](#) (const [DSMatrix](#) \*lvalue, const [DSMatrix](#) \*rvalue)
- [DSMatrix](#) \* [DSMatrixByApplyingFunction](#) (const [DSMatrix](#) \*mvalue, double(\*function)(double))
- [DSMatrix](#) \* [DSMatrixBySubtractingScalar](#) (const [DSMatrix](#) \*lvalue, const double rvalue)
- [DSMatrix](#) \* [DSMatrixByAddingScalar](#) (const [DSMatrix](#) \*lvalue, const double rvalue)
- [DSMatrix](#) \* [DSMatrixByDividingScalar](#) (const [DSMatrix](#) \*lvalue, const double rvalue)
- [DSMatrix](#) \* [DSMatrixByMultiplyingScalar](#) (const [DSMatrix](#) \*lvalue, const double rvalue)
- double [DSMatrixDoubleValue](#) (const [DSMatrix](#) \*matrix, const DSUInteger row, const DSUInteger column)  
*Returns the element of the [DSMatrix](#) specified by a row and column.*
- void [DSMatrixSetDoubleValue](#) ([DSMatrix](#) \*matrix, const DSUInteger row, const DSUInteger column, const double value)
- void [DSMatrixSetDoubleValuesList](#) ([DSMatrix](#) \*matrix, bool byColumns, DSUInteger numberOfValues, double firstValue,...)
- void [DSMatrixSetDoubleValues](#) ([DSMatrix](#) \*matrix, bool byColumns, DSUInteger numberOfValues, double \*values)
- void [DSMatrixSetDoubleValueAll](#) ([DSMatrix](#) \*matrix, const double value)  
*Sets all the values of a matrix to a value.*
- void [DSMatrixRoundToSignificantFigures](#) ([DSMatrix](#) \*matrix, const unsigned char figures)
- [DSMatrix](#) \* [DSMatrixSubMatrixExcludingRowList](#) (const [DSMatrix](#) \*matrix, const DSUInteger numberOfRows, const DSUInteger firstRow,...)

- **DSMatrix \* DSMatrixSubMatrixExcludingRows** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger \*rows)
- **DSMatrix \* DSMatrixSubMatrixExcludingColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRowsColumns, const DSUInteger firstColumn,...)
- **DSMatrix \* DSMatrixSubMatrixExcludingColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfRowsColumns, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixSubMatrixIncludingRowList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger firstRow,...)
- **DSMatrix \* DSMatrixSubMatrixIncludingRows** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger \*rows)
- **DSMatrix \* DSMatrixSubMatrixIncludingColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRowsColumns, const DSUInteger firstColumn,...)
- **DSMatrix \* DSMatrixSubMatrixIncludingColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfRowsColumns, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixSubMatrixExcludingRowAndColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfRowsColumns, const DSUInteger firstRow,...)
- **DSMatrix \* DSMatrixSubMatrixExcludingRowsAndColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfRowsColumns, const DSUInteger \*rows, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixSubMatrixIncludingRowAndColumnList** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfRowsColumns, const DSUInteger firstRow,...)
- **DSMatrix \* DSMatrixSubMatrixIncludingRowsAndColumns** (const **DSMatrix** \*matrix, const DSUInteger numberOfRows, const DSUInteger numberOfRowsColumns, const DSUInteger \*rows, const DSUInteger \*columns)
- **DSMatrix \* DSMatrixAppendMatrices** (const **DSMatrix** \*firstMatrix, const **DSMatrix** \*secondMatrix, const bool byColumn)
- void **DSMatrixSwitchRows** (**DSMatrix** \*matrix, const DSUInteger rowA, const DSUInteger rowB)
- void **DSMatrixSwitchColumns** (**DSMatrix** \*matrix, const DSUInteger columnA, const DSUInteger columnB)
- **DSMatrix \* DSMatrixWithUniqueRows** (const **DSMatrix** \*matrix)
- void **DSMatrixPrint** (const **DSMatrix** \*matrix)
- bool **DSMatrixIsIdentity** (const **DSMatrix** \*matrix)
- bool **DSMatrixIsSquare** (const **DSMatrix** \*matrix)
- DSUInteger **DSMatrixRank** (const **DSMatrix** \*matrix)
- double **minimumValue** (const **DSMatrix** \*matrix, const bool shouldExcludeZero)
- double **maximumValue** (const **DSMatrix** \*matrix, const bool shouldExcludeZero)
- void **DSMatrixAddByMatrix** (**DSMatrix** \*addTo, const **DSMatrix** \*addBy)
- void **DSMatrixSubtractByMatrix** (**DSMatrix** \*addTo, const **DSMatrix** \*addBy)
- void **DSMatrixApplyFunction** (**DSMatrix** \*matrix, double(\*function)(double))
- void **DSMatrixMultiplyByScalar** (**DSMatrix** \*matrix, const double value)
- double **DSMatrixDeterminant** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixTranspose** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixInverse** (const **DSMatrix** \*matrix)
- **DSMatrixArray \* DSMatrixSVD** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixRightNullspace** (const **DSMatrix** \*matrix)
- **DSMatrix \* DSMatrixLeftNullspace** (const **DSMatrix** \*matrix)
- **DSMatrixArray \* DSMatrixPLUdecomposition** (const **DSMatrix** \*A)  
*Creates a LU decomposition and returns the permutation matrix.*
- double \* **DSMatrixDataForGLPK** (const **DSMatrix** \*matrix)
- int \* **DSMatrixRowsForGLPK** (const **DSMatrix** \*matrix)
- int \* **DSMatrixColumnsForGLPK** (const **DSMatrix** \*matrix)

### 7.16.1 Detailed Description

Implementation file with functions for dealing with matrices using the GNU Scientific Library (gsl). Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.16.2 Function Documentation

#### 7.16.2.1 DSMatrix\* DSMatrixAlloc (const DSUInteger *rows*, const DSUInteger *columns*)

Memory allocation for a [DSMatrix](#) using malloc.

Creates a new matrix of a particular size. The matrix that is allocated has all the values of the matrix defaulted to 0. The internal matrix pointer must be set to NULL; otherwise, the size of the matrix cannot be changed.

#### Parameters

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

#### Returns

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

#### 7.16.2.2 DSMatrix\* DSMatrixByAddingMatrix (const DSMatrix \* *lvalue*, const DSMatrix \* *rvalue*)

Create a new [DSMatrix](#) object by adding a matrix to another.

This function takes two matrices of the same dimensions, and adds the ij element of the rvalue matrix to the ij element of the lvalue matrix. This function assumes constant matrices, and thus does not modify either of the inputs, but instead creates a copy of the first operand matrix, and calls DSMatrixAddByMatrix(), using the copy as the first operand.

#### Parameters

*lvalue* The first [DSMatrix](#) object to be added.

*rvalue* The second [DSMatrix](#) object to be added.

### Returns

If the addition operation was succesful, the function returns a pointer to the newly allocated matrix. Otherwise, NULL is returned.

### See also

[DSMatrixAddByMatrix\(\)](#)

#### 7.16.2.3 [DSMatrix\\*](#) [DSMatrixByParsingString](#) (const char \* *string*)

Creates a new matrix by parsing a tab-delimited matrix.

This function reads an input string, containing rows delimited by tabs and columns delimited by newlines. This function generates a token stream, and thus checks the dimensions of the matrix prior to creating it.

### Parameters

*string* A string containing the data to parse.

### Returns

A [DSMatrix](#) data object with the parsed data. If parsing failed, returns NULL.

#### 7.16.2.4 [DSMatrix\\*](#) [DSMatrixBySubtractingMatrix](#) (const [DSMatrix](#) \* *lvalue*, const [DSMatrix](#) \* *rvalue*)

Create a new [DSMatrix](#) object by subtracting a matrix from another.

This function takes two matrices of the same dimensions, and subtracts the ij element of the rvalue matrix to the ij element of the lvalue matrix. This function assumes constant matrices, and thus does not modify either of the inputs, but instead creates a copy of the minuend operand matrix, and called [DSMatrixSubtractByMatrix\(\)](#) with the copy as the new minuend.

### Parameters

*lvalue* The [DSMatrix](#) object that is the minuend.

*rvalue* The [DSMatrix](#) object that is the subtrahend.

### Returns

If the subtraction operation was succesful, the function returns a pointer to the newly allocated difference matrix. Otherwise, NULL is returned.

### See also

[DSMatrixSubtractByMatrix\(\)](#)

### 7.16.2.5 DSMatrix\* DSMatrixCalloc (const DSUInteger rows, const DSUInteger columns)

Memory allocation for a [DSMatrix](#) using calloc.

Creates a new matrix of a particular size. The matrix that is allocated has all the values of the matrix defaulted to 0. The internal matrix pointer must be set to NULL; otherwise, the size of the matrix cannot be changed.

#### Parameters

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

#### Returns

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

### 7.16.2.6 DSMatrix\* DSMatrixCopy (const DSMatrix \* original)

Copies a [DSMatrix](#).

Creates a new matrix with the exact same size and contents as some other matrix. The new matrix is allocated, and thus must be freed.

#### Parameters

*original* The [DSMatrix](#) to be copied.

#### Returns

If the copy was succesful, a pointer to a copy of the [DSMatrix](#) is returned. Otherwise, NULL is returned.

### 7.16.2.7 double DSMatrixDoubleValue (const DSMatrix \* matrix, const DSUInteger row, const DSUInteger column)

Returns the element of the [DSMatrix](#) specified by a row and column.

Returns an element of the matrix, with indices i and j starting at 0.

#### Parameters

*matrix* The [DSMatrix](#) whose elements will be accessed.

*row* A DSUInteger specifying the row coordinate of the element to be accessed.

*column* A DSUInteger specifying the column coordinate of the element to be accessed.

#### Returns

If the value was succesfully retrieved, the double value contained at the row and column coordinate of the [DSMatrix](#) is returned. Otherwise, NaN is returned.

**7.16.2.8 void DSMatrixFree (DSMatrix \* *matrix*)**

Freeing memory for [DSMatrix](#).

Frees the memory associated with a [DSMatrix](#) data type. This function is a wrapper for the necessary steps needed to free the internal structure of the [DSMatrix](#) data type.

**Parameters**

*matrix* The [DSMatrix](#) to be freed.

**7.16.2.9 DSMatrix\* DSMatrixIdentity (const DSUInteger *size*)**

Allocates a new [DSMatrix](#) as an identity matrix.

Allocates a square matrix of a specified size, and initializes the diagonal values to 1 and all the other values to 0, creating an identity matrix. The new matrix is therefore an identity matrix.

**Parameters**

*size* A DSUInteger containing the number of rows and columns in the matrix.

**Returns**

If the identity matrix was successfully created, a pointer to the [DSMatrix](#) is returned. Otherwise, NULL is returned.

**7.16.2.10 DSMatrixArray\* DSMatrixPLUdecomposition (const DSMatrix \* *A*)**

Creates a LU decomposition and returns the permutation matrix.

This function creates a LU decomposition of a [DSMatrix](#) *A*. This function creates an array of three matrices: a [DSMatrix](#) *P*, a [DSMatrix](#) *L* and a [DSMatrix](#) *U*; where  $PA = LU$ .

**Parameters**

*A* A [DSMatrix](#) containing the matrix to be decomposed.

**7.16.2.11 DSMatrix\* DSMatrixRandomNumbers (const DSUInteger *rows*, const DSUInteger *columns*)**

Allocates a new [DSMatrix](#) with random values between 0 and 1.

Allocates a new [DSMatrix](#) with a specified size. The values of each of the entries in the matrix are randomly selected between 0 and 1.

**Parameters**

*rows* A DSUInteger with the number of rows in the new matrix.

*columns* A DSUInteger with the number of columns in the new matrix.

**Returns**

If the matrix was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

**7.16.2.12 void DSMatrixSetDoubleValueAll (DSMatrix \* *matrix*, const double *value*)**

Sets all the values of a matrix to a value.

This function does not allocate the necessary memory; instead it goes through all the rows and columns of the matrix, assigning them the specified value.

**Parameters**

*matrix* The [DSMatrix](#) that will be assigned the value.

*value* The double variable whose value will be assigned.





This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.17.2 Function Documentation

#### 7.17.2.1 void DSMatrixArrayAddMatrix (DSMatrixArray \* array, const DSMatrix \* matrixToAdd)

Function to add a new matrix to the [DSMatrixArray](#).

This function is the standard mechanism to add a [DSMatrix](#) to a [DSMatrixArray](#). This function allocates the necessary space in the internal C array, and adds the [DSMatrix](#) to the end of the array. Once added to the matrix array, the memory is managed by the matrix array and is freed upon calling DSMatrixArrayFree.

#### Parameters

*array* The [DSMatrixArray](#) that will have a new matrix added.

*matrixToAdd* The [DSMatrix](#) to be added to the matrix array.

#### 7.17.2.2 DSMatrixArray\* DSMatrixArrayAlloc (void)

Memory allocation for a [DSMatrixArray](#).

Creates a new [DSMatrixArray](#) with no matrices. As matrices are added, the matrix array grows, therefore the matrix array is initialized to 0, with a NULL internal pointer and number of matrices set to 0.

#### Returns

If the matrix array was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

#### 7.17.2.3 DSMatrixArray\* DSMatrixArrayCopy (const DSMatrixArray \* array)

Copies a [DSMatrixArray](#).

Creates a new [DSMatrixArray](#) with the exact same data and contents as some other matrix array. The matrices in the new [DSMatrixArray](#) are copies of the matrices in the original matrix array.

**Parameters**

*array* The [DSMatrixArray](#) to be copied.

**Returns**

If the copy was succesful, a pointer to a copy of the [DSMatrixArray](#) is returned. Otherwise, NULL is returned.

**See also**

[DSMatrixCopy](#)

**7.17.2.4 void DSMatrixArrayFree (DSMatrixArray \* *array*)**

Freeing memory for [DSMatrixArray](#).

Frees the memory associated with a [DSMatrixArray](#) data type. This function is a wrapper for the necessary steps needed to free the internal structure of the [DSMatrixArray](#), this includes calling [DSMatrixFree](#) for each of the contained matrices, freeing the internal pointer to the array of matrices, and the [DSMatrixArray](#) data type itself.

**Parameters**

*array* The [DSMatrixArray](#) to be freed.

**7.17.2.5 DSMatrix\* DSMatrixArrayMatrix (const DSMatrixArray \* *array*, const DSUInteger *index*)**

Function to access a matrix in the [DSMatrixArray](#).

This accessor function returns the [DSMatrix](#) at the specified index of the [DSMatrixArray](#). This function is the basic accessor function, and should always be used to access a matrix in a [DSMatrixArray](#).

**Parameters**

*array* The [DSMatrixArray](#) containing the matrix to be accessed.

*index* The DSUInteger specifying the index in the C array of matrices contained in the [DSMatrixArray](#).

**Returns**

If the [DSMatrix](#) at the specified index was found, the pointer to that matrix is returned. Otherwise, NULL is returned.

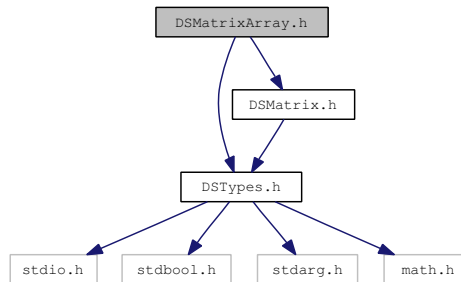
## 7.18 DSMatrixArray.h File Reference

Header file with functions for dealing with matrix arrays.

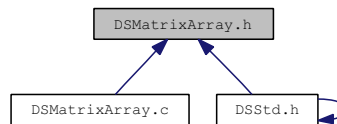
```
#include "DSTypes.h"
```

```
#include "DSMatrix.h"
```

Include dependency graph for DSMatrixArray.h:



This graph shows which files directly or indirectly include this file:



### Defines

- `#define DSMatrixArrayNumberOfMatrices(x) ((x)->numberOfMatrices)`  
*Accessor function to retrieve number of matrices in the Matrix array.*
- `#define DSMatrixArrayInternalPointer(x) ((x)->matrices)`  
*Accessor function to retrieve the pointer to the C matrix array.*

### Functions

- `DSMatrixArray * DSMatrixArrayAlloc (void)`  
*Memory allocation for a DSMatrixArray.*
- `DSMatrixArray * DSMatrixArrayCopy (const DSMatrixArray *array)`  
*Copies a DSMatrixArray.*
- `void DSMatrixArrayFree (DSMatrixArray *array)`  
*Freeing memory for DSMatrixArray.*
- `DSMatrix * DSMatrixArrayMatrix (const DSMatrixArray *array, const DSUInteger index)`  
*Function to access a matrix in the DSMatrixArray.*

- void [DSMatrixArrayAddMatrix](#) ([DSMatrixArray](#) \*array, const [DSMatrix](#) \*matrixToAdd)  
*Function to add a new matrix to the [DSMatrixArray](#).*
- double [DSMatrixArrayDoubleWithIndices](#) (const [DSMatrixArray](#) \*array, const DSUInteger i, const DSUInteger j, const DSUInteger k)
- void [DSMatrixArrayPrint](#) (const [DSMatrixArray](#) \*array)

### 7.18.1 Detailed Description

Header file with functions for dealing with matrix arrays. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.18.2 Function Documentation

#### 7.18.2.1 void [DSMatrixArrayAddMatrix](#) ([DSMatrixArray](#) \* *array*, const [DSMatrix](#) \* *matrixToAdd*)

Function to add a new matrix to the [DSMatrixArray](#).

This function is the standard mechanism to add a [DSMatrix](#) to a [DSMatrixArray](#). This function allocates the necessary space in the internal C array, and adds the [DSMatrix](#) to the end of the array. Once added to the matrix array, the memory is managed by the matrix array and is freed upon calling [DSMatrixArrayFree](#).

#### Parameters

*array* The [DSMatrixArray](#) that will have a new matrix added.

*matrixToAdd* The [DSMatrix](#) to be added to the matrix array.

#### 7.18.2.2 [DSMatrixArray](#)\* [DSMatrixArrayAlloc](#) (void)

Memory allocation for a [DSMatrixArray](#).

Creates a new [DSMatrixArray](#) with no matrices. As matrices are added, the matrix array grows, therefore the matrix array is initialized to 0, with a NULL internal pointer and number of matrices set to 0.

**Returns**

If the matrix array was created, a new pointer to a [DSMatrix](#) is returned. Otherwise, NULL is returned.

**7.18.2.3 DSMatrixArray\* DSMatrixArrayCopy (const DSMatrixArray \* array)**

Copies a [DSMatrixArray](#).

Creates a new [DSMatrixArray](#) with the exact same data and contents as some other matrix array. The matrices in the new [DSMatrixArray](#) are copies of the matrices in the original matrix array.

**Parameters**

*array* The [DSMatrixArray](#) to be copied.

**Returns**

If the copy was succesful, a pointer to a copy of the [DSMatrixArray](#) is returned. Otherwise, NULL is returned.

**See also**

[DSMatrixCopy](#)

**7.18.2.4 void DSMatrixArrayFree (DSMatrixArray \* array)**

Freeing memory for [DSMatrixArray](#).

Frees the memory associated with a [DSMatrixArray](#) data type. This function is a wrapper for the necessary steps needed to free the internal structure of the [DSMatrixArray](#), this includes calling [DSMatrixFree](#) for each of the contained matrices, freeing the internal pointer to the array of matrices, and the [DSMatrixArray](#) data type itself.

**Parameters**

*array* The [DSMatrixArray](#) to be freed.

**7.18.2.5 DSMatrix\* DSMatrixArrayMatrix (const DSMatrixArray \* array, const DSUInteger *index*)**

Function to access a matrix in the [DSMatrixArray](#).

This accessor function returns the [DSMatrix](#) at the specified index of the [DSMatrixArray](#). This function is the basic accessor function, and should always be used to access a matrix in a [DSMatrixArray](#).

**Parameters**

*array* The [DSMatrixArray](#) containing the matrix to be accessed.

*index* The DSUInteger specifying the index in the C array of matrices contained in the [DSMatrixArray](#).

**Returns**

If the [DSMatrix](#) at the specified index was found, the pointer to that matrix is returned. Otherwise, NULL is returned.

## 7.19 DSMatrixTokenizer.c File Reference

Implementation file with functions for tokenizing with matrices.

```
#include <stdio.h>
```

```
#include "DSMatrixTokenizer.h"
```

Include dependency graph for DSMatrixTokenizer.c:

### Functions

- struct [matrix\\_token](#) \* **DSMatrixTokenAlloc** ()
- void **DSMatrixTokenFree** (struct [matrix\\_token](#) \*root)

### 7.19.1 Detailed Description

Implementation file with functions for tokenizing with matrices. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.20 DSMatrixTokenizer.h File Reference

Header file with functions for tokenizing matrices.

```
#include "DSTypes.h"
#include "DSErrors.h"
#include "DSMemoryManager.h"
```

Include dependency graph for DSMatrixTokenizer.h: This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [matrix\\_token](#)

### Defines

- #define [DS\\_MATRIX\\_TOKEN\\_START](#) 0  
*Token indicating the start of a tokenization.*
- #define [DS\\_MATRIX\\_TOKEN\\_DOUBLE](#) 1  
*Token indicating a numerical value.*
- #define [DS\\_MATRIX\\_TOKEN\\_NEWLINE](#) 2  
*Token indicating a newline, indicative of a new row.*
- #define [DS\\_MATRIX\\_TOKEN\\_ERROR](#) 3  
*Token indicating an error during tokenization.*
- #define [DSMatrixTokenNext](#)(x) ((x)->next)
- #define [DSMatrixTokenValue](#)(x) ((x)->value)
- #define [DSMatrixTokenType](#)(x) ((x)->token)
- #define [DSMatrixTokenRow](#)(x) ((x)->row)
- #define [DSMatrixTokenColumn](#)(x) ((x)->column)
- #define [DSMatrixTokenSetNext](#)(x, y) ((x)->next = (y))
- #define [DSMatrixTokenSetValue](#)(x, y) ((x)->value = (y))
- #define [DSMatrixTokenSetType](#)(x, y) ((x)->token = (y))
- #define [DSMatrixTokenSetRow](#)(x, y) ((x)->row = (y))
- #define [DSMatrixTokenSetColumn](#)(x, y) ((x)->column = (y))

### Functions

- struct [matrix\\_token](#) \* [DSMatrixTokenAlloc](#) ()
- void [DSMatrixTokenFree](#) (struct [matrix\\_token](#) \*root)
- struct [matrix\\_token](#) \* [DSMatrixTokenizeString](#) (const char \*string)



### 7.20.1 Detailed Description

Header file with functions for tokenizing matrices. This header file specifies the data structure relating to the tokenization of an input string to be parsed as a matrix, as well as all the functions necessary to tokenize it. This file is a private file, and therefore its contents will be invisible to the public API. As such, it is not necessary to place the C++ compatibility declaration.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

This header file specifies the data structure relating to the tokenization of an input string to be parsed as a matrix, as well as all the functions necessary to tokenize it. This file is a private file, and therefore its contents will be invisible to the public API. Therefore, it is unnecessary to place the C++ compatibility declarations.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.21 DSMatrixTokenizerLex.c File Reference

Implementation file with functions for tokenizing matrices, generated by flex.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSMemoryManager.h"
#include "DSMatrix.h"
#include "DSMatrixTokenizer.h"
#include <unistd.h>
```

Include dependency graph for DSMatrixTokenizerLex.c:

### Data Structures

- struct [yy\\_buffer\\_state](#)
- struct [yy\\_trans\\_info](#)
- struct [yyguts\\_t](#)

### Defines

- #define **YY\_INT\_ALIGNED** short int
- #define **FLEX\_SCANNER**
- #define **YY\_FLEX\_MAJOR\_VERSION** 2
- #define **YY\_FLEX\_MINOR\_VERSION** 5
- #define **YY\_FLEX\_SUBMINOR\_VERSION** 35
- #define **INT16\_MIN** (-32767-1)
- #define **INT32\_MIN** (-2147483647-1)
- #define **INT8\_MAX** (127)
- #define **INT16\_MAX** (32767)
- #define **INT32\_MAX** (2147483647)
- #define **UINT8\_MAX** (255U)
- #define **UINT16\_MAX** (65535U)
- #define **UINT32\_MAX** (4294967295U)
- #define **yyconst**
- #define **YY\_NULL** 0
- #define **YY\_SC\_TO\_UI**(c) ((unsigned int) (unsigned char) c)
- #define **YY\_TYPEDEF\_Y\_Y\_SCANNER\_T**
- #define **yyin** yyg->yyin\_r
- #define **yyout** yyg->yyout\_r
- #define **yyextra** yyg->yyextra\_r
- #define **yylen** yyg->yylen\_r
- #define **yytext** yyg->yytext\_r
- #define **yylineno** (YY\_CURRENT\_BUFFER\_LVALUE->yy\_bs\_lineno)

- `#define yycolumn (YY_CURRENT_BUFFER_LVALUE->yy_bs_column)`
- `#define yy_flex_debug yyg->yy_flex_debug_r`
- `#define BEGIN yyg->yy_start = 1 + 2 *`
- `#define YY_START ((yyg->yy_start - 1) / 2)`
- `#define YYSTATE YY_START`
- `#define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)`
- `#define YY_NEW_FILE DSMatrixFlexrestart(yyin ,yyscanner )`
- `#define YY_END_OF_BUFFER_CHAR 0`
- `#define YY_BUF_SIZE 16384`
- `#define YY_STATE_BUF_SIZE ((YY_BUF_SIZE + 2) * sizeof(yy_state_type))`
- `#define YY_TYPEDEF_YY_BUFFER_STATE`
- `#define YY_TYPEDEF_YY_SIZE_T`
- `#define EOB_ACT_CONTINUE_SCAN 0`
- `#define EOB_ACT_END_OF_FILE 1`
- `#define EOB_ACT_LAST_MATCH 2`
- `#define YY_LESS_LINENO(n)`
- `#define yyless(n)`
- `#define unput(c) yyunput( c, yyg->ytext_ptr , yyscanner )`
- `#define YY_STRUCT_YY_BUFFER_STATE`
- `#define YY_BUFFER_NEW 0`
- `#define YY_BUFFER_NORMAL 1`
- `#define YY_BUFFER_EOF_PENDING 2`
- `#define YY_CURRENT_BUFFER`
- `#define YY_CURRENT_BUFFER_LVALUE yyg->yy_buffer_stack[yyg->yy_buffer_stack_top]`
- `#define YY_FLUSH_BUFFER DSMatrixFlex_flush_buffer(YY_CURRENT_BUFFER ,yyscan-`  
ner)
- `#define yy_new_buffer DSMatrixFlex_create_buffer`
- `#define yy_set_interactive(is_interactive)`
- `#define yy_set_bol(at_bol)`
- `#define YY_AT_BOL() (YY_CURRENT_BUFFER_LVALUE->yy_at_bol)`
- `#define ytext_ptr ytext_r`
- `#define YY_DO_BEFORE_ACTION`
- `#define YY_NUM_RULES 9`
- `#define YY_END_OF_BUFFER 10`
- `#define REJECT reject_used_but_not_detected`
- `#define ymore() ymore_used_but_not_detected`
- `#define YY_MORE_ADJ 0`
- `#define YY_RESTORE_YY_MORE_OFFSET`
- `#define malloc(x) DSSecureMalloc(x)`
- `#define calloc(x, y) DSSecureCalloc(x, y)`
- `#define realloc(x, y) DSSecureRealloc(x, y)`
- `#define INITIAL 0`
- `#define YY_EXTRA_TYPE struct matrix\_token *`
- `#define YY_READ_BUF_SIZE 8192`
- `#define ECHO fwrite( ytext, yyleng, 1, yyout )`
- `#define YY_INPUT(buf, result, max_size)`
- `#define yyterminate() return YY_NULL`
- `#define YY_START_STACK_INCR 25`
- `#define YY_FATAL_ERROR(msg) yy_fatal_error( msg , yyscanner)`

- `#define YY_DECL_IS_OURS 1`
- `#define YY_DECL int DSMatrixFlexlex (yyscan_t yyscanner)`
- `#define YY_USER_ACTION`
- `#define YY_BREAK break;`
- `#define YY_RULE_SETUP YY_USER_ACTION`
- `#define YY_EXIT_FAILURE 2`
- `#define yyless(n)`
- `#define YYTABLES_NAME "yytables"`

## Typedefs

- typedef signed char `flex_int8_t`
- typedef short int `flex_int16_t`
- typedef int `flex_int32_t`
- typedef unsigned char `flex_uint8_t`
- typedef unsigned short int `flex_uint16_t`
- typedef unsigned int `flex_uint32_t`
- typedef void \* `yyscan_t`
- typedef struct `yy_buffer_state` \* `YY_BUFFER_STATE`
- typedef size\_t `yy_size_t`
- typedef unsigned char `YY_CHAR`
- typedef int `yy_state_type`

## Functions

- void `DSMatrixFlexrestart` (FILE \*input\_file, yyscan\_t yyscanner)
- void `DSMatrixFlex_switch_to_buffer` (`YY_BUFFER_STATE` new\_buffer, yyscan\_t yyscanner)
- `YY_BUFFER_STATE` `DSMatrixFlex_create_buffer` (FILE \*file, int size, yyscan\_t yyscanner)
- void `DSMatrixFlex_delete_buffer` (`YY_BUFFER_STATE` b, yyscan\_t yyscanner)
- void `DSMatrixFlex_flush_buffer` (`YY_BUFFER_STATE` b, yyscan\_t yyscanner)
- void `DSMatrixFlexpush_buffer_state` (`YY_BUFFER_STATE` new\_buffer, yyscan\_t yyscanner)
- void `DSMatrixFlexpop_buffer_state` (yyscan\_t yyscanner)
- `YY_BUFFER_STATE` `DSMatrixFlex_scan_buffer` (char \*base, yy\_size\_t size, yyscan\_t yyscanner)
- `YY_BUFFER_STATE` `DSMatrixFlex_scan_string` (yyconst char \*yy\_str, yyscan\_t yyscanner)
- `YY_BUFFER_STATE` `DSMatrixFlex_scan_bytes` (yyconst char \*bytes, yy\_size\_t len, yyscan\_t yyscanner)
- void \* `DSMatrixFlexalloc` (yy\_size\_t, yyscan\_t yyscanner)
- void \* `DSMatrixFlexrealloc` (void \*, yy\_size\_t, yyscan\_t yyscanner)
- void `DSMatrixFlexfree` (void \*, yyscan\_t yyscanner)
- int `DSMatrixFlexlex_init` (yyscan\_t \*scanner)
- int `DSMatrixFlexlex_init_extra` (`YY_EXTRA_TYPE` user\_defined, yyscan\_t \*scanner)
- int `DSMatrixFlexlex_destroy` (yyscan\_t yyscanner)
- int `DSMatrixFlexget_debug` (yyscan\_t yyscanner)
- void `DSMatrixFlexset_debug` (int debug\_flag, yyscan\_t yyscanner)
- `YY_EXTRA_TYPE` `DSMatrixFlexget_extra` (yyscan\_t yyscanner)
- void `DSMatrixFlexset_extra` (`YY_EXTRA_TYPE` user\_defined, yyscan\_t yyscanner)
- FILE \* `DSMatrixFlexget_in` (yyscan\_t yyscanner)
- void `DSMatrixFlexset_in` (FILE \*in\_str, yyscan\_t yyscanner)
- FILE \* `DSMatrixFlexget_out` (yyscan\_t yyscanner)

- void **DSMatrixFlexset\_out** (FILE \*out\_str, yyscan\_t yyscanner)
- yy\_size\_t **DSMatrixFlexget\_leng** (yyscan\_t yyscanner)
- char \* **DSMatrixFlexget\_text** (yyscan\_t yyscanner)
- int **DSMatrixFlexget\_lineno** (yyscan\_t yyscanner)
- void **DSMatrixFlexset\_lineno** (int line\_number, yyscan\_t yyscanner)
- int **DSMatrixFlexwrap** (yyscan\_t yyscanner)
- int **DSMatrixFlexlex** (yyscan\_t yyscanner)
- int **DSMatrixFlexget\_column** (yyscan\_t yyscanner)
- void **DSMatrixFlexset\_column** (int column\_no, yyscan\_t yyscanner)
- struct **matrix\_token** \* **DSMatrixTokenizeString** (const char \*string)

### 7.21.1 Detailed Description

Implementation file with functions for tokenizing matrices, generated by flex. This file was generated directly by the flex program, and is the source code responsible for matrix tokenization. This file was generated by flex, according to a specification written by Jason Lomnitz. To generate this file, the following command must be executed: "flex -t DSMatrixGrammar.l > DSMatrixTokenizerLex.c".

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.21.2 Define Documentation

#### 7.21.2.1 #define YY\_CURRENT\_BUFFER

##### Value:

```
( yyg->yy_buffer_stack \
    ? yyg->yy_buffer_stack[yyg->yy_buffer_stack_top] \
    : NULL)
```

#### 7.21.2.2 #define YY\_DO\_BEFORE\_ACTION

##### Value:

```

yyg->yytext_ptr = yy_bp; \
  yylen = (size_t) (yy_cp - yy_bp); \
  yyg->yy_hold_char = *yy_cp; \
  *yy_cp = '\0'; \
  yyg->yy_c_buf_p = yy_cp;

```

### 7.21.2.3 #define YY\_INPUT(buf, result, max\_size)

**Value:**

```

if ( YY_CURRENT_BUFFER_LVALUE->yy_is_interactive ) \
{ \
  int c = '*'; \
  yy_size_t n; \
  for ( n = 0; n < max_size && \
        (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
    buf[n] = (char) c; \
  if ( c == '\n' ) \
    buf[n++] = (char) c; \
  if ( c == EOF && ferror( yyin ) ) \
    YY_FATAL_ERROR( "input in flex scanner failed" ); \
  result = n; \
} \
else \
{ \
  errno=0; \
  while ( (result = fread(buf, 1, max_size, yyin))==0 && ferror(yyi
n)) \
  { \
    if( errno != EINTR) \
    { \
      YY_FATAL_ERROR( "input in flex scanner failed" ); \
      break; \
    } \
    errno=0; \
    clearerr(yyin); \
  } \
} \
\

```

### 7.21.2.4 #define yy\_set\_bol(at\_bol)

**Value:**

```

{ \
  if ( ! YY_CURRENT_BUFFER ){ \
    DSMatrixFlexensure_buffer_stack (yyscanner); \
    YY_CURRENT_BUFFER_LVALUE = \
      DSMatrixFlex_create_buffer(yyin,YY_BUF_SIZE ,yyscanner); \
  } \
  YY_CURRENT_BUFFER_LVALUE->yy_at_bol = at_bol; \
}

```

### 7.21.2.5 #define yy\_set\_interactive(is\_interactive)

**Value:**

```

{ \

```

```

if ( ! YY_CURRENT_BUFFER ){ \
DSMatrixFlexensure_buffer_stack (yyscanner); \
    YY_CURRENT_BUFFER_LVALUE = \
        DSMatrixFlex_create_buffer(yyin,YY_BUF_SIZE ,yyscanner); \
} \
YY_CURRENT_BUFFER_LVALUE->yy_is_interactive = is_interactive; \
}

```

### 7.21.2.6 #define yyless(n)

**Value:**

```

do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        yytext[yylen] = yyg->yy_hold_char; \
        yyg->yy_c_buf_p = yytext + yyless_macro_arg; \
        yyg->yy_hold_char = *yyg->yy_c_buf_p; \
        *yyg->yy_c_buf_p = '\0'; \
        yylen = yyless_macro_arg; \
    } \
while ( 0 )

```

### 7.21.2.7 #define yyless(n)

**Value:**

```

do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO(yyless_macro_arg);\
        *yy_cp = yyg->yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yyg->yy_c_buf_p = yy_cp = yy_bp + yyless_macro_arg - YY_MORE_ADJ; \
        YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
while ( 0 )

```

## 7.21.3 Function Documentation

### 7.21.3.1 void DSMatrixFlex\_flush\_buffer (YY\_BUFFER\_STATE *b*, yyscan\_t *yyscanner*)

Discard all buffered characters. On the next scan, YY\_INPUT will be called.

**Parameters**

- b*** the buffer state to be flushed, usually YY\_CURRENT\_BUFFER.
- yyscanner*** The scanner object.

### 7.21.3.2 YY\_BUFFER\_STATE DSMatrixFlex\_scan\_buffer (char \**base*, yy\_size\_t *size*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan directly from a user-specified character buffer.

**Parameters**

*base* the character buffer  
*size* the size in bytes of the character buffer  
*yyscanner* The scanner object.

**Returns**

the newly allocated buffer state object.

**7.21.3.3 YY\_BUFFER\_STATE DSMatrixFlex\_scan\_bytes (yyconst char \* yybytes, yy\_size\_t \_yybytes\_len, yyscan\_t yyscanner)**

Setup the input buffer state to scan the given bytes. The next call to DSMatrixFlexlex() will scan from a *copy* of *bytes*.

**Parameters**

*bytes* the byte buffer to scan  
*len* the number of bytes in the buffer pointed to by *bytes*.  
*yyscanner* The scanner object.

**Returns**

the newly allocated buffer state object.

**7.21.3.4 YY\_BUFFER\_STATE DSMatrixFlex\_scan\_string (yyconst char \* yystr, yyscan\_t yyscanner)**

Setup the input buffer state to scan a string. The next call to DSMatrixFlexlex() will scan from a *copy* of *str*.

**Parameters**

*yystr* a NUL-terminated string to scan  
*yyscanner* The scanner object.

**Returns**

the newly allocated buffer state object.

**Note**

If you want to scan bytes that may contain NUL values, then use [DSMatrixFlex\\_scan\\_bytes\(\)](#) instead.

**7.21.3.5 int DSMatrixFlexget\_column (yyscan\_t yyscanner)**

Get the current column number.

**Parameters**

*yyscanner* The scanner object.



**7.21.3.6 YY\_EXTRA\_TYPE DSMatrixFlexget\_extra (yyscan\_t *yyscanner*)**

Get the user-defined data for this scanner.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.7 FILE \* DSMatrixFlexget\_in (yyscan\_t *yyscanner*)**

Get the input stream.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.8 yy\_size\_t DSMatrixFlexget\_leng (yyscan\_t *yyscanner*)**

Get the length of the current token.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.9 int DSMatrixFlexget\_lineno (yyscan\_t *yyscanner*)**

Get the current line number.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.10 FILE \* DSMatrixFlexget\_out (yyscan\_t *yyscanner*)**

Get the output stream.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.11 char \* DSMatrixFlexget\_text (yyscan\_t *yyscanner*)**

Get the current token.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.12 void DSMatrixFlexpop\_buffer\_state (yyvsp\_t yyscanner)**

Removes and deletes the top of the stack, if present. The next element becomes the new top.

**Parameters**

*yyscanner* The scanner object.

**7.21.3.13 void DSMatrixFlexpush\_buffer\_state (YY\_BUFFER\_STATE *new\_buffer*, yyscan\_t yyscanner)**

Pushes the new state onto the stack. The new state becomes the current state. This function will allocate the stack if necessary.

**Parameters**

*new\_buffer* The new state.

*yyscanner* The scanner object.

**7.21.3.14 void DSMatrixFlexset\_column (int *column\_no*, yyscan\_t yyscanner)**

Set the current column.

**Parameters**

*line\_number*

*yyscanner* The scanner object.

**7.21.3.15 void DSMatrixFlexset\_extra (YY\_EXTRA\_TYPE *user\_defined*, yyscan\_t yyscanner)**

Set the user-defined data. This data is never touched by the scanner.

**Parameters**

*user\_defined* The data to be associated with this scanner.

*yyscanner* The scanner object.

**7.21.3.16 void DSMatrixFlexset\_in (FILE \* *in\_str*, yyscan\_t yyscanner)**

Set the input stream. This does not discard the current input buffer.

**Parameters**

*in\_str* A readable stream.

*yyscanner* The scanner object.

**See also**

DSMatrixFlex\_switch\_to\_buffer

**7.21.3.17 void DSMatrixFlexset\_lineno (int *line\_number*, yyscan\_t *yyscanner*)**

Set the current line number.

**Parameters**

*line\_number*

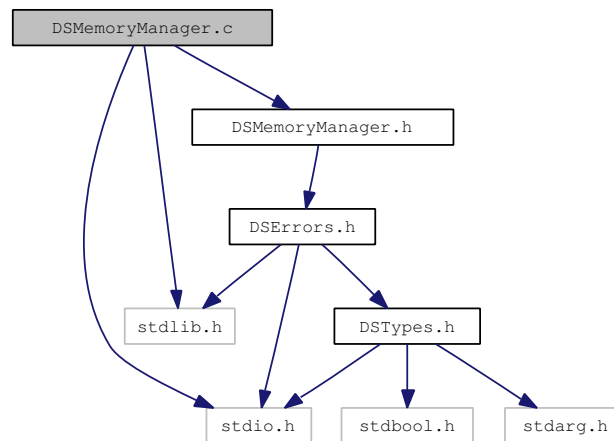
*yyscanner* The scanner object.

## 7.22 DSMemoryManager.c File Reference

implementation file with functions for secure memory management.

```
#include <stdio.h>
#include <stdlib.h>
#include "DSMemoryManager.h"
```

Include dependency graph for DSMemoryManager.c:



### Functions

- void \* [DSSecureMalloc](#) (size\_t size)  
*Function to securely allocate data using malloc.*
- void \* [DSSecureCalloc](#) (size\_t count, size\_t size)  
*Function to securely allocate data using calloc.*
- void \* [DSSecureRealloc](#) (void \*ptr, size\_t size)  
*Function to securely allocate data using realloc.*
- void [DSSecureFree](#) (void \*ptr)  
*Function to securely free data.*

### 7.22.1 Detailed Description

implementation file with functions for secure memory management. This file specifies the design space standard for error handling. Contained here are the necessary macros and functions to successfully report the errors throughout the design space library.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

**7.22.2 Function Documentation****7.22.2.1 void\* DSSecureCalloc (size\_t count, size\_t size)**

Function to securely allocate data using calloc.

This function is a secure calloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

A pointer to the allocated data.

**7.22.2.2 void DSSecureFree (void \* ptr)**

Function to securely free data.

This function is a secure free function which checks the data pointer. If the data pointer is null, indicative of errors when freeing memory, the function issues a fatal error. This function calls malloc in case that pointer to be reallocated is NULL.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

A pointer to the allocated data.

**7.22.2.3 void\* DSSecureMalloc (size\_t size)**

Function to securely allocate data using malloc.

This function is a secure malloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error.

**Parameters**

*size* A DSUInteger specifying the size of memory being allocated.

**Returns**

A pointer to the allocated data.

**7.22.2.4 void\* DSSecureRealloc (void \* ptr, size\_t size)**

Function to securely allocate data using realloc.

This function is a secure realloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error. This function calls malloc in case that pointer to be reallocated is NULL.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

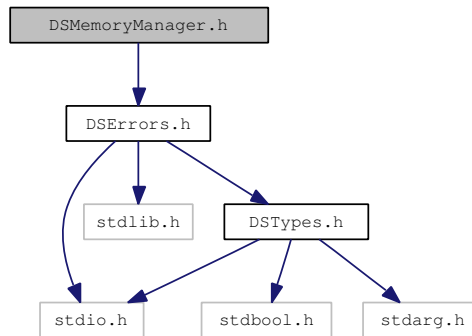
A pointer to the allocated data.

## 7.23 DSMemoryManager.h File Reference

Header file with functions for secure memory allocation.

```
#include "DSErrors.h"
```

Include dependency graph for DSMemoryManager.h:



This graph shows which files directly or indirectly include this file:

### Functions

- void \* [DSSecureMalloc](#) (size\_t size)  
*Function to securely allocate data using malloc.*
- void \* [DSSecureCalloc](#) (size\_t count, size\_t size)  
*Function to securely allocate data using calloc.*
- void \* [DSSecureRealloc](#) (void \*ptr, size\_t size)  
*Function to securely allocate data using realloc.*
- void [DSSecureFree](#) (void \*ptr)  
*Function to securely free data.*

### 7.23.1 Detailed Description

Header file with functions for secure memory allocation. This file specifies the design space standard for error handling. Contained here are the necessary macros and functions to successfully report the errors throughout the design space library.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

## 7.23.2 Function Documentation

### 7.23.2.1 void\* DSSecureCalloc (size\_t count, size\_t size)

Function to securely allocate data using calloc.

This function is a secure calloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

A pointer to the allocated data.

### 7.23.2.2 void DSSecureFree (void \* ptr)

Function to securely free data.

This function is a secure free function which checks the data pointer. If the data pointer is null, indicative of errors when freeing memory, the function issues a fatal error. This function calls malloc in case that pointer to be reallocated is NULL.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

A pointer to the allocated data.

### 7.23.2.3 void\* DSSecureMalloc (size\_t size)

Function to securely allocate data using malloc.

This function is a secure malloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error.



**Parameters**

*size* A DSUInteger specifying the size of memory being allocated.

**Returns**

A pointer to the allocated data.

**7.23.2.4 void\* DSSecureRealloc (void \* *ptr*, size\_t *size*)**

Function to securely allocate data using realloc.

This function is a secure realloc function which checks the allocated pointer. If the data pointer is null, indicative of errors allocating memory, the function issues a fatal error. This function calls malloc in case that pointer to be reallocated is NULL.

**Parameters**

*count* A DSUInteger specifying the number of memory blocks being allocated.

*size* The memory size of each block being allocated.

**Returns**

A pointer to the allocated data.

## 7.24 DSSSystem.h File Reference

Header file with functions for dealing with S-System.

```
#include "DSTypes.h"
```

Include dependency graph for DSSSystem.h: This graph shows which files directly or indirectly include this file:

### Defines

- `#define M_DS_SSYS_NULL M_DS_NULL` ": S-System is NULL"

### Functions

- `void DSSSystemFree (DSSSystem *ssys)`
- `__deprecated DSSSystem * DSSSystemFromGMAWithDominantTerms (const DSGMASystem *gma, const DSUInteger *termList)`
- `DSSSystem * DSSSystemWithTermsFromGMA (const DSGMASystem *gma, const DSUInteger *termArray)`
- `DSSSystem * DSSSystemByParsingStringList (const DSVariablePool *Xi0, const char *const string,...)`
- `DSSSystem * DSSSystemByParsingStrings (const DSVariablePool *Xi0, char *const *const strings, const DSUInteger numberOfEquations)`
- `double DSSSystemSteadyStateFunction (const DSSSystem *ssys, const DSVariablePool *Xi0, const char *function)`
- `DSMatrix * DSSSystemSteadyStateValues (const DSSSystem *ssys, const DSVariablePool *Xi0)`
- `DSMatrix * DSSSystemSteadyStateFlux (const DSSSystem *ssys, const DSVariablePool *Xi0)`
- `const DSUInteger DSSSystemNumberOfEquations (const DSSSystem *ssys)`
- `DSExpression ** DSSSystemEquations (const DSSSystem *ssys)`
- `DSExpression ** DSSSystemSolution (const DSSSystem *ssys)`
- `DSExpression ** DSSSystemLogarithmicSolution (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemAlpha (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemBeta (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemGd (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemGi (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemHd (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemHi (const DSSSystem *ssys)`
- `const DSMatrix * DSSSystemM (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemAd (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemAi (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemB (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemA (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemG (const DSSSystem *ssys)`
- `DSMatrix * DSSSystemH (const DSSSystem *ssys)`
- `const DSVariablePool * DSSSystemXd (const DSSSystem *const ssys)`
- `const DSVariablePool * DSSSystemXi (const DSSSystem *const ssys)`
- `const bool DSSSystemHasSolution (const DSSSystem *ssys)`
- `const bool DSSSystemIsSingular (const DSSSystem *ssys)`
- `void DSSSystemPrint (const DSSSystem *ssys)`
- `void DSSSystemPrintEquations (const DSSSystem *ssys)`
- `void DSSSystemPrintSolution (const DSSSystem *ssys)`
- `void DSSSystemPrintLogarithmicSolution (const DSSSystem *ssys)`

### 7.24.1 Detailed Description

Header file with functions for dealing with S-System. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.25 DSStd.h File Reference

Header file for the design space toolbox.

```
#include <stdio.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSIO.h"
#include "DSErrors.h"
#include "DSMemoryManager.h"
#include "DSVariable.h"
#include "DSMatrix.h"
#include "DSMatrixArray.h"
#include "DSExpression.h"
#include "DSGMASystem.h"
#include "DSSSystem.h"
#include "DSCase.h"
#include "DSDesignSpace.h"
#include "DSVertices.h"
#include "DSDictionary.h"
#include "DSStack.h"
```

Include dependency graph for DSStd.h:

### Defines

- #define **free**(x) DSSecureFree(x)
- #define **malloc**(x) DSSecureMalloc(x)
- #define **calloc**(x, y) DSSecureCalloc(x, y)
- #define **realloc**(x, y) DSSecureRealloc(x, y)

### 7.25.1 Detailed Description

Header file for the design space toolbox. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

2011

**Todo**

Add all previous functionality.  
Add vertex enumeration functionality.

## 7.26 DSSymbolicMatrix.h File Reference

Header file with functions for dealing with symbolic matrices.

```
#include "DSTypes.h"
#include "DSErrors.h"
#include "DSIO.h"
```

Include dependency graph for DSSymbolicMatrix.h:

### Defines

- #define [M\\_DS\\_SYM\\_MAT\\_NULL](#) "Pointer to symbolic matrix is NULL"  
*Message for a NULL DSMatrix pointer.*
- #define [M\\_DS\\_SYM\\_MAT\\_OUTOFBOUNDS](#) "Row or column out of bounds"  
*Message for a row or column exceeding matrix bounds.*
- #define [M\\_DS\\_SYM\\_MAT\\_NOINTERNAL](#) "Matrix data is empty"  
*Message for a NULL internal matrix structure.*

### Functions

- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixAlloc](#) (const DSUInteger rows, const DSUInteger columns)
- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixCalloc](#) (const DSUInteger rows, const DSUInteger columns)
- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixCopy](#) (const [DSSymbolicMatrix](#) \*original)
- void [DSSymbolicMatrixFree](#) ([DSSymbolicMatrix](#) \*matrix)
- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixIdentity](#) (const DSUInteger size)
- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixRandomNumbers](#) (const DSUInteger rows, const DSUInteger columns)
- [DSSymbolicMatrix](#) \* [DSSymbolicMatrixByParsingString](#) (const char \*string)
- double [DSSymbolicMatrixDoubleByEvaluatingExpression](#) (const [DSSymbolicMatrix](#) \*matrix, const DSUInteger row, const DSUInteger column, const [DSVariablePool](#) \*variableValues)
- const [DSExpression](#) \* [DSSymbolicMatrixExpression](#) (const [DSSymbolicMatrix](#) \*matrix, const DSUInteger row, const DSUInteger column)
- void [DSSymbolicMatrixSetExpression](#) ([DSSymbolicMatrix](#) \*matrix, const DSUInteger row, const DSUInteger column, const [DSExpression](#) \*expr)
- DSUInteger [DSSymbolicMatrixRows](#) (const [DSSymbolicMatrix](#) \*matrix)
- DSUInteger [DSSymbolicMatrixColumns](#) (const [DSSymbolicMatrix](#) \*matrix)
- [DSMatrix](#) \* [DSSymbolicMatrixToNumericalMatrix](#) (const [DSSymbolicMatrix](#) \*matrix, const [DSVariablePool](#) \*variables)

### 7.26.1 Detailed Description

Header file with functions for dealing with symbolic matrices. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

**Author**

Jason Lomnitz.

**Date**

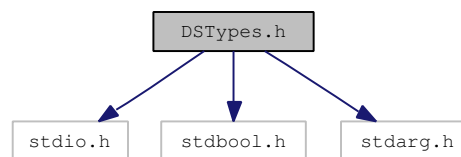
2011

## 7.27 DTypes.h File Reference

Header file with definitions for data types.

```
#include <stdio.h>
#include <stdbool.h>
#include <stdarg.h>
#include <math.h>
#include <complex.h>
#include <pthread.h>
```

Include dependency graph for DTypes.h:



This graph shows which files directly or indirectly include this file:

### Data Structures

- struct [DSVertices](#)  
*Data type that contains vertices of an N-Dimensional object.*
- struct [DSVariable](#)  
*Basic variable structure containing name, value and NSString with special unicode characters for greek letters.*
- struct [\\_varDictionary](#)  
*Internal dictionary structure.*
- struct [DSDictionary](#)
- struct [DSVariablePool](#)  
*User-level variable pool.*
- struct [dsexpression](#)  
*Data type representing mathematical expressions.*
- struct [DSSymbolicMatrix](#)  
*Data type representing a symbolic matrix.*
- struct [DSMatrix](#)  
*Data type representing a matrix.*
- struct [DSMatrixArray](#)  
*Data type representing an array of matrices.*



- struct [DSGMASystem](#)  
*Data type representing a GMA-System.*
- struct [DSSSystem](#)  
*Data type representing an S-System.*
- struct [DSCase](#)  
*Data type used to represent a case.*
- struct [DSStack](#)
- struct [DSDesignSpace](#)  
*Data type used to represent a design space/.*

## Defines

- #define **endif**
- #define **\_\_deprecated**
- #define **INFINITY** HUGE\_VAL

## Typedefs

- typedef int **DSInteger**
- typedef unsigned int **DSUInteger**
- typedef struct [\\_varDictionary](#) [DSInternalDictionary](#)  
*Internal dictionary structure.*
- typedef struct [dsexpression](#) [DSExpression](#)  
*Data type representing mathematical expressions.*
- typedef [DSMatrix](#) [DSComplexMatrix](#)  
*Data type representing a matrix with complex values.*

## Enumerations

- enum [DSVariablePoolLock](#) { [DSLockReadWriteAdd](#), [DSLockReadWrite](#), [DSLockReadOnly](#), [DSLockLocked](#) }  
*Data type used to lock different properties of the [DSVariablePool](#).*

### 7.27.1 Detailed Description

Header file with definitions for data types. This file specifies the design space standard data types. Contained here are strictly the data type definitions. Functions applying to these data types are contained elsewhere, and the individual data structures should refer to the respective files.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see [<http://www.gnu.org/licenses/>](http://www.gnu.org/licenses/).

**Author**

Jason Lomnitz.

**Date**

2011

## 7.27.2 Typedef Documentation

### 7.27.2.1 typedef DSMatrix DSComplexMatrix

Data type representing a matrix with complex values.

This data type is the front end of the matrix manipulation portion of the design space toolbox. Currently, the DST library uses the gsl library; however, it is designed to be used with different back-ends. In particular, the CLAPACK package should be considered, as it will offer better performance. Thus, the matrix API should be independent of implementation, and hence a new matrix library could be used if chosen.

**See also**

DSComplexMatrix.h  
DSComplexMatrix.c

### 7.27.2.2 typedef struct dsexpression DSExpression

Data type representing mathematical expressions.

This data type is the internal representation of mathematical expressions. This data type is an Abstracts Syntax Tree with only three operators: '+', '\*', and '^'. All other operators ('-' and '/') are represented by a combination of the former operators. The DSExpression automatically groups constant values, and reserves the first branch of the multiplication and addition operator for constant values. These operators can have any number of branches. The '^' operator can have two, and only two, branches.

**Note**

Functions are handled as variables with a single argument

**See also**

[DSExpression.h](#)  
[DSExpression.c](#)

### 7.27.2.3 typedef struct \_varDictionary DSInternalDictionary

Internal dictionary structure.

Internal dictionary for fast variable querying. The structure of the dictionary uses an alternative path, where each character is checked in order at each position, if there is a match, the next position is consequently checked. The dictionary should never be manipulated manually, adding, retrieving and removing variables should be done through the accessory functions.

See also

[DSDictionary](#)

## 7.27.3 Enumeration Type Documentation

### 7.27.3.1 enum DSVariablePoolLock

Data type used to lock different properties of the [DSVariablePool](#).

This data type enumerates the properties of the variable pool access rights. Its values indicate the different operations that can be taken with a variable pool, such as read/write/add, read/write and read.

See also

[DSVariable.h](#)

[DSVariable.c](#)

Enumerator:

***DSLockReadWriteAdd*** The value of the Variable pool lock indicating read/write/add.

***DSLockReadWrite*** The value of the Variable pool lock indicating read/write.

***DSLockReadOnly*** The value of the Variable pool lock indicating read/.

***DSLockLocked*** The value of the Variable pool lock indicating no access.

## 7.28 DSVariable.c File Reference

Implementation file with functions the DSInternalDictionary object.

```
#include <stdbool.h>
#include <string.h>
#include <math.h>
#include <pthread.h>
#include "DSMemoryManager.h"
#include "DSErrors.h"
#include "DSVariable.h"
#include "DSVariableTokenizer.h"
#include "DSTypes.h"
#include "DSMatrix.h"
```

Include dependency graph for DSVariable.c: This graph shows which files directly or indirectly include this file:

### Defines

- #define **dsVarDictionarySetValue**(x, y) ((x != NULL) ? x->value = y : DSError(M\_DS\_WRONG ": Dictionary is NULL", A\_DS\_ERROR))
- #define **dsVarDictionaryValue**(x) ((x != NULL) ? x->value : NULL)
- #define **dsVariablePoolNumberOfVariables**(x) ((x)->numberOfVariables)

### Functions

- **DSVariable** \* **DSVariableAlloc** (const char \*name)  
*Creates a new DSVariable with INFINITY as a default value.*
- void **DSVariableFree** (DSVariable \*var)  
*Function frees allocated memory of a DSVariable.*
- void **DSVariablePrint** (const DSVariable \*var)
- **DSVariable** \* **DSVariableRetain** (DSVariable \*aVariable)  
*Function to increase variable retain count by one.*
- void **DSVariableRelease** (DSVariable \*aVariable)  
*Function to decrease variable retain count by one.*
- **DSVariablePool** \* **DSVariablePoolAlloc** (void)  
*Creates a new DSVariablePool with an empty var dictionary.*
- **DSVariablePool** \* **DSVariablePoolCopy** (const DSVariablePool \*const reference)  
*Creates a new DSVariablePool with a copy of the reference variable pool.*
- void **DSVariablePoolFree** (DSVariablePool \*pool)

*Creates a new [DSVariablePool](#) with a copy of the reference variable pool.*

- void [DSVariablePoolSetReadOnly](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read only.*
- void [DSVariablePoolSetReadWrite](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read and write.*
- void [DSVariablePoolSetReadWriteAdd](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read, write and add.*
- DSUInteger [DSVariablePoolNumberOfVariables](#) (const [DSVariablePool](#) \*pool)  
*Function to retrieve the number of variables in a [DSVariablePool](#).*
- bool [DSVariablePoolIsReadOnly](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read only.*
- bool [DSVariablePoolIsReadWrite](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read and write.*
- bool [DSVariablePoolIsReadWriteAdd](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read, write and add.*
- void [DSVariablePoolAddVariableWithName](#) ([DSVariablePool](#) \*pool, const char \*name)  
*Creates and adds a new variable to the variable pool.*
- void [DSVariablePoolAddVariable](#) ([DSVariablePool](#) \*pool, [DSVariable](#) \*newVar)  
*Adds an existing variable to the variable pool.*
- bool [DSVariablePoolHasVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*const name)  
*Checks if a [DSVariablePool](#) has a variable with a specified name.*
- [DSVariable](#) \* [DSVariablePoolVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name)
- void [DSVariablePoolSetValueForVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name, const double value)
- const [DSVariable](#) \*\* [DSVariablePoolAllVariables](#) (const [DSVariablePool](#) \*pool)
- const char \*\* [DSVariablePoolAllVariableNames](#) (const [DSVariablePool](#) \*pool)
- DSUInteger [DSVariablePoolIndexOfVariable](#) (const [DSVariablePool](#) \*pool, const [DSVariable](#) \*var)
- DSUInteger [DSVariablePoolIndexOfVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name)
- [DSVariablePool](#) \* [DSVariablePoolByParsingString](#) (const char \*string)
- void [DSVariablePoolPrint](#) (const [DSVariablePool](#) \*const pool)
- [DSMatrix](#) \* [DSVariablePoolValuesAsVector](#) (const [DSVariablePool](#) \*pool, const bool rowVector)

## Variables

- pthread\_mutex\_t **retaincount**

### 7.28.1 Detailed Description

Implementation file with functions the DSInternalDictionary object. Implementation file with functions for dealing with variables.

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.28.2 Function Documentation

#### 7.28.2.1 DSVariable\* DSVariableAlloc (const char \* *name*)

Creates a new [DSVariable](#) with INFINITY as a default value.

This function may be used throughout, in order to create new variables consistently and portably. As variables are allocated individually, it is important to not that they should be released with the accessory method.

#### Parameters

*name* A string with which to identify the [DSVariable](#).

#### Returns

The pointer to the newly allocated [DSVariable](#).

#### See also

[DSVariable](#)  
[DSVariableFree](#)

#### 7.28.2.2 void DSVariableFree (DSVariable \* *var*)

Function frees allocated memory of a [DSVariable](#).

This function should not be used explicitly, as the [DSVariable](#) object has an internal memory counter. This function is ultimately called when the variable memory counter reaches zero. Freeing a [DSVariable](#) object should be done through the DSVariableRelease function, and never should a [DSVariable](#) be directly freed, as its internal structure may be subject to future changes.

### Parameters

*var* The pointer to the variable to free.

### See also

[DSVariableRetain\(\)](#)  
[DSVariableRelease\(\)](#)

#### 7.28.2.3 void DSVariablePoolAddVariable (DSVariablePool \* *pool*, DSVariable \* *newVar*)

Adds an existing variable to the variable pool.

This function acts on an existing [DSVariablePool](#) object, adding an existing variable with a specified name to the internal dictionary structure. The variable added is not created, but this function calls [DSVariableRetain](#), thus increasing the memory retain count of the variable by one. If a variable already exists with the same name, this function does not add the variable to the pool, and throws a warning.

### Parameters

*pool* The [DSVariablePool](#) object to which a new variable will be added.  
*name* A null terminated string with the name of the variable to add.

### See also

[DSVariablePoolAddVariableWithName\(\)](#)  
[DSVariableRetain\(\)](#)

#### 7.28.2.4 void DSVariablePoolAddVariableWithName (DSVariablePool \* *pool*, const char \* *name*)

Creates and adds a new variable to the variable pool.

This function acts on an existing [DSVariablePool](#) object, creating a new variable with a specified name and adding it to the internal dictionary structure. If a variable already exists with the same name, this function does not create a new variable, and throws a warning.

### Parameters

*pool* The [DSVariablePool](#) object to which a new variable will be added.  
*name* A null terminated string with the name of the variable to add.

#### 7.28.2.5 DSVariablePool\* DSVariablePoolAlloc (void)

Creates a new [DSVariablePool](#) with an empty var dictionary.

The variable pool is initialized with read/write privileges. The variable pool stores a indexed version of the variables added, as well as the order in which the variables were added. The order of the variables is kept to ensure a consistent variable index with system matrices of S-Systems and GMAs.

### Returns

The pointer to the allocated [DSVariablePool](#).

### See also

[DSVariablePoolFree](#)

#### 7.28.2.6 DSVariablePool\* DSVariablePoolCopy (const DSVariablePool \*const *reference*)

Creates a new [DSVariablePool](#) with a copy of the reference variable pool.

The variable pool that is created is initialized with the same read/write/add privileges as the reference variable pool. The contents of the variable pool are an exact copy of the reference variable pool. Despite the contents being the same, the variables in each pool are independent, thus new variables are created in the copy.

##### Parameters

*reference* A [DSVariablePool](#) data type that serves as the reference variable pool, which is to be copied.

##### Returns

The copy of the reference [DSVariablePool](#) object (must be freed by user).

##### See also

[DSVariablePoolFree\(\)](#)

#### 7.28.2.7 void DSVariablePoolFree (DSVariablePool \* *pool*)

Creates a new [DSVariablePool](#) with a copy of the reference variable pool.

The variable pool that is created is initialized with the same read/write/add privileges as the reference variable pool. The contents of the variable pool are an exact copy of the reference variable pool. Despite the contents being the same, the variables in each pool are independent, thus new variables are created in the copy.

##### Parameters

*reference* A [DSVariablePool](#) data type that serves as the reference variable pool, which is to be copied.

##### Returns

The copy of the reference [DSVariablePool](#) object (must be freed by user).

##### See also

[DSVariablePoolFree\(\)](#)

#### 7.28.2.8 bool DSVariablePoolIsReadOnly (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read only.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read only.

##### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

##### See also

[DSVariablePoolIsReadWrite\(\)](#)  
[DSVariablePoolIsReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)



### 7.28.2.9 bool DSVariablePoolIsReadWrite (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read and write.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read and write.

#### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

#### See also

[DSVariablePoolIsReadOnly\(\)](#)  
[DSVariablePoolIsReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

### 7.28.2.10 bool DSVariablePoolIsReadWriteAdd (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read, write and add.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read, write and add.

#### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

#### See also

[DSVariablePoolIsReadOnly\(\)](#)  
[DSVariablePoolIsReadWrite\(\)](#)  
[DSVariablePoolLock](#)

### 7.28.2.11 DSUInteger DSVariablePoolNumberOfVariables (const DSVariablePool \* *pool*)

Function to retrieve the number of variables in a [DSVariablePool](#).

#### Parameters

*pool* A [DSVariablePool](#) object that to query its number of variables.

### 7.28.2.12 void DSVariablePoolSetReadOnly (DSVariablePool \* *pool*)

Changes the existing privileges of a [DSVariablePool](#) object to read only.

This function acts on an existing [DSVariablePool](#) object, and changes the existing privileges to read-only. This privilege setting prohibits adding new variables to the variable pool, or changing the value of a variable explicitly. The value of a variable can be changed directly, but not through the variable pool interface.

#### Parameters

*pool* A [DSVariablePool](#) object that will have its privileges changed.

#### See also

[DSVariablePoolSetReadWrite\(\)](#)  
[DSVariablePoolSetReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

**7.28.2.13 void DSVariablePoolSetReadWrite (DSVariablePool \* *pool*)**

Changes the existing privileges of a [DSVariablePool](#) object to read and write.

This function acts on an existing [DSVariablePool](#) object, and changes its privileges to read and write. This privilege setting prohibits adding new variables to the variable pool. The value of a variable can be changed through the variable pool interface.

**Parameters**

*pool* A [DSVariablePool](#) object that will have its privileges changed.

**See also**

[DSVariablePoolSetReadOnly\(\)](#)  
[DSVariablePoolSetReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

**7.28.2.14 void DSVariablePoolSetReadWriteAdd (DSVariablePool \* *pool*)**

Changes the existing privileges of a [DSVariablePool](#) object to read, write and add.

This function acts on an existing [DSVariablePool](#) object, and changes its privileges to read, write and add. This privilege setting allows adding new variables to the variable pool and changing the values of the variables.

**Parameters**

*pool* A [DSVariablePool](#) object that will have its privileges changed.

**See also**

[DSVariablePoolSetReadOnly\(\)](#)  
[DSVariablePoolSetReadWrite\(\)](#)  
[DSVariablePoolLock](#)

**7.28.2.15 void DSVariableRelease (DSVariable \* *aVariable*)**

Function to decrease variable retain count by one.

[DSVariable](#) object is made to decrease its retain count by one, when the retain count hits zero, the function [DSVariableFree\(\)](#) is invoked, freeing the memory of the [DSVariable](#) object. [DSVariable](#) objects do not have an equivalent to autorelease, forcing the developer to invoke a [DSRelease](#) for each [DSRetain](#) explicitly called.

**Parameters**

*aVariable* The variable which will have its retain count reduced.

**See also**

[DSVariableRetain](#)  
[DSVariableFree](#)

**7.28.2.16 DSVariable\* DSVariableRetain (DSVariable \* *aVariable*)**

Function to increase variable retain count by one.

Variables utilize a similar memory management system used in Objective-C NSObject subclasses. A [DSVariable](#) recently allocated begins with a retain count of one.

**Parameters**

*aVariable* The variable which will have its retain count increased.

**Returns**

The same variable which received the retain count increase is returned, for convinience.

**See also**

[DSVariableRelease](#)

## 7.29 DSVariable.h File Reference

Header file with functions for dealing with variables.

```
#include <stdio.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSDictionary.h"
```

Include dependency graph for DSVariable.h: This graph shows which files directly or indirectly include this file:

### Defines

- #define **DSVariableAssignValue**(x, y) DSVariableSetValue(x, y)
- #define **DSVariableReturnValue**(x) DSVariableValue(x)
- #define **DSVariableSetValue**(x, y) (((DSVariable\*)(x))->value = (y))  
*Macro to set the value of a variable data structure.*
- #define **DSVariableValue**(x) (((x) != NULL) ? ((DSVariable\*)(x))->value : NAN)  
*Macro to get the value of a variable data structure.*
- #define **DSVariableName**(x) (((DSVariable\*)(x))->name)  
*Macro to get the value of a variable data structure.*
- #define **M\_DS\_VAR\_NULL** M\_DS\_NULL ": Variable Pool is NULL"  
*Error message indicating a NULL variable pool.*
- #define **M\_DS\_VAR\_LOCKED** " DSVariablePool: Insufficient privileges"  
*Error message indicating insufficient privileges to manipulate a variable pool.*
- #define **DSVariablePoolInternalDictionary**(x) ((x)->dictionary)
- #define **DSVariablePoolVariableArray**(x) ((x)->variables)

### Functions

- **DSVariable \*** **DSVariableAlloc** (const char \*name)  
*Creates a new DSVariable with INFINITY as a default value.*
- void **DSVariableFree** (DSVariable \*var)  
*Function frees allocated memory of a DSVariable.*
- **DSVariable \*** **DSVariableRetain** (DSVariable \*aVariable)  
*Function to increase variable retain count by one.*
- void **DSVariableRelease** (DSVariable \*aVariable)  
*Function to decrease variable retain count by one.*
- **DSVariablePool \*** **DSVariablePoolAlloc** (void)

*Creates a new [DSVariablePool](#) with an empty var dictionary.*

- [DSVariablePool](#) \* [DSVariablePoolCopy](#) (const [DSVariablePool](#) \*const pool)  
*Creates a new [DSVariablePool](#) with a copy of the reference variable pool.*
- void [DSVariablePoolFree](#) ([DSVariablePool](#) \*pool)  
*Creates a new [DSVariablePool](#) with a copy of the reference variable pool.*
- [DSVariablePool](#) \* [DSVariablePoolByParsingString](#) (const char \*string)
- void [DSVariablePoolSetReadOnly](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read only.*
- void [DSVariablePoolSetReadWrite](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read and write.*
- void [DSVariablePoolSetReadWriteAdd](#) ([DSVariablePool](#) \*pool)  
*Changes the existing privileges of a [DSVariablePool](#) object to read, write and add.*
- void [DSVariablePoolAddVariableWithName](#) ([DSVariablePool](#) \*pool, const char \*name)  
*Creates and adds a new variable to the variable pool.*
- void [DSVariablePoolAddVariable](#) ([DSVariablePool](#) \*pool, [DSVariable](#) \*newVar)  
*Adds an existing variable to the variable pool.*
- void [DSVariablePoolSetValueForVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name, const double value)
- DSUInteger [DSVariablePoolNumberOfVariables](#) (const [DSVariablePool](#) \*pool)  
*Function to retrieve the number of variables in a [DSVariablePool](#).*
- bool [DSVariablePoolIsReadOnly](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read only.*
- bool [DSVariablePoolIsReadWrite](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read and write.*
- bool [DSVariablePoolIsReadWriteAdd](#) (const [DSVariablePool](#) \*pool)  
*Queries the existing privileges of a [DSVariablePool](#) object, checking it is read, write and add.*
- bool [DSVariablePoolHasVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*const name)  
*Checks if a [DSVariablePool](#) has a variable with a specified name.*
- [DSVariable](#) \* [DSVariablePoolVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name)
- const [DSVariable](#) \*\* [DSVariablePoolAllVariables](#) (const [DSVariablePool](#) \*pool)
- const char \*\* [DSVariablePoolAllVariableNames](#) (const [DSVariablePool](#) \*pool)
- DSUInteger [DSVariablePoolIndexOfVariable](#) (const [DSVariablePool](#) \*pool, const [DSVariable](#) \*var)
- DSUInteger [DSVariablePoolIndexOfVariableWithName](#) (const [DSVariablePool](#) \*pool, const char \*name)
- void [DSVariablePoolPrint](#) (const [DSVariablePool](#) \*const pool)
- [DSMatrix](#) \* [DSVariablePoolValuesAsVector](#) (const [DSVariablePool](#) \*pool, const bool rowVector)

### 7.29.1 Detailed Description

Header file with functions for dealing with variables. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

### 7.29.2 Function Documentation

#### 7.29.2.1 DSVariable\* DSVariableAlloc (const char \* *name*)

Creates a new [DSVariable](#) with INFINITY as a default value.

This function may be used throughout, in order to create new variables consistently and portably. As variables are allocated individually, it is important to not that they should be released with the accessory method.

#### Parameters

*name* A string with which to identify the [DSVariable](#).

#### Returns

The pointer to the newly allocated [DSVariable](#).

#### See also

[DSVariable](#)  
[DSVariableFree](#)

#### 7.29.2.2 void DSVariableFree (DSVariable \* *var*)

Function frees allocated memory of a [DSVariable](#).

This function should not be used explicitly, as the [DSVariable](#) object has an internal memory counter. This function is ultimately called when the variable memory counter reaches zero. Freeing a [DSVariable](#) object should be done through the [DSVariableRelease](#) function, and never should a [DSVariable](#) be directly freed, as its internal structure may be subject to future changes.

### Parameters

*var* The pointer to the variable to free.

### See also

[DSVariableRetain\(\)](#)  
[DSVariableRelease\(\)](#)

#### 7.29.2.3 void DSVariablePoolAddVariable (DSVariablePool \* *pool*, DSVariable \* *newVar*)

Adds an existing variable to the variable pool.

This function acts on an existing [DSVariablePool](#) object, adding an existing variable with a specified name to the internal dictionary structure. The variable added is not created, but this function calls [DSVariableRetain](#), thus increasing the memory retain count of the variable by one. If a variable already exists with the same name, this function does not add the variable to the pool, and throws a warning.

### Parameters

*pool* The [DSVariablePool](#) object to which a new variable will be added.  
*name* A null terminated string with the name of the variable to add.

### See also

[DSVariablePoolAddVariableWithName\(\)](#)  
[DSVariableRetain\(\)](#)

#### 7.29.2.4 void DSVariablePoolAddVariableWithName (DSVariablePool \* *pool*, const char \* *name*)

Creates and adds a new variable to the variable pool.

This function acts on an existing [DSVariablePool](#) object, creating a new variable with a specified name and adding it to the internal dictionary structure. If a variable already exists with the same name, this function does not create a new variable, and throws a warning.

### Parameters

*pool* The [DSVariablePool](#) object to which a new variable will be added.  
*name* A null terminated string with the name of the variable to add.

#### 7.29.2.5 DSVariablePool\* DSVariablePoolAlloc (void)

Creates a new [DSVariablePool](#) with an empty var dictionary.

The variable pool is initialized with read/write privileges. The variable pool stores a indexed version of the variables added, as well as the order in which the variables were added. The order of the variables is kept to ensure a consistent variable index with system matrices of S-Systems and GMAs.

### Returns

The pointer to the allocated [DSVariablePool](#).

### See also

[DSVariablePoolFree](#)

### 7.29.2.6 DSVariablePool\* DSVariablePoolCopy (const DSVariablePool \*const *reference*)

Creates a new [DSVariablePool](#) with a copy of the reference variable pool.

The variable pool that is created is initialized with the same read/write/add privileges as the reference variable pool. The contents of the variable pool are an exact copy of the reference variable pool. Despite the contents being the same, the variables in each pool are independent, thus new variables are created in the copy.

#### Parameters

*reference* A [DSVariablePool](#) data type that serves as the reference variable pool, which is to be copied.

#### Returns

The copy of the reference [DSVariablePool](#) object (must be freed by user).

#### See also

[DSVariablePoolFree\(\)](#)

### 7.29.2.7 void DSVariablePoolFree (DSVariablePool \* *pool*)

Creates a new [DSVariablePool](#) with a copy of the reference variable pool.

The variable pool that is created is initialized with the same read/write/add privileges as the reference variable pool. The contents of the variable pool are an exact copy of the reference variable pool. Despite the contents being the same, the variables in each pool are independent, thus new variables are created in the copy.

#### Parameters

*reference* A [DSVariablePool](#) data type that serves as the reference variable pool, which is to be copied.

#### Returns

The copy of the reference [DSVariablePool](#) object (must be freed by user).

#### See also

[DSVariablePoolFree\(\)](#)

### 7.29.2.8 bool DSVariablePoolIsReadOnly (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read only.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read only.

#### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

#### See also

[DSVariablePoolIsReadWrite\(\)](#)  
[DSVariablePoolIsReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)



### 7.29.2.9 bool DSVariablePoolIsReadWrite (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read and write.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read and write.

#### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

#### See also

[DSVariablePoolIsReadOnly\(\)](#)  
[DSVariablePoolIsReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

### 7.29.2.10 bool DSVariablePoolIsReadWriteAdd (const DSVariablePool \* *pool*)

Queries the existing privileges of a [DSVariablePool](#) object, checking it is read, write and add.

This function acts on an existing [DSVariablePool](#) object, and checks if its privileges are read, write and add.

#### Parameters

*pool* A [DSVariablePool](#) object to be queried for its privileges.

#### See also

[DSVariablePoolIsReadOnly\(\)](#)  
[DSVariablePoolIsReadWrite\(\)](#)  
[DSVariablePoolLock](#)

### 7.29.2.11 DSUInteger DSVariablePoolNumberOfVariables (const DSVariablePool \* *pool*)

Function to retrieve the number of variables in a [DSVariablePool](#).

#### Parameters

*pool* A [DSVariablePool](#) object that to query its number of variables.

### 7.29.2.12 void DSVariablePoolSetReadOnly (DSVariablePool \* *pool*)

Changes the existing privileges of a [DSVariablePool](#) object to read only.

This function acts on an existing [DSVariablePool](#) object, and changes the existing privileges to read-only. This privilege setting prohibits adding new variables to the variable pool, or changing the value of a variable explicitly. The value of a variable can be changed directly, but not through the variable pool interface.

#### Parameters

*pool* A [DSVariablePool](#) object that will have its privileges changed.

#### See also

[DSVariablePoolSetReadWrite\(\)](#)  
[DSVariablePoolSetReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

**7.29.2.13 void DSVariablePoolSetReadWrite (DSVariablePool \* *pool*)**

Changes the existing privileges of a [DSVariablePool](#) object to read and write.

This function acts on an existing [DSVariablePool](#) object, and changes its privileges to read and write. This privilege setting prohibits adding new variables to the variable pool. The value of a variable can be changed through the variable pool interface.

**Parameters**

*pool* A [DSVariablePool](#) object that will have its privileges changed.

**See also**

[DSVariablePoolSetReadOnly\(\)](#)  
[DSVariablePoolSetReadWriteAdd\(\)](#)  
[DSVariablePoolLock](#)

**7.29.2.14 void DSVariablePoolSetReadWriteAdd (DSVariablePool \* *pool*)**

Changes the existing privileges of a [DSVariablePool](#) object to read, write and add.

This function acts on an existing [DSVariablePool](#) object, and changes its privileges to read, write and add. This privilege setting allows adding new variables to the variable pool and changing the values of the variables.

**Parameters**

*pool* A [DSVariablePool](#) object that will have its privileges changed.

**See also**

[DSVariablePoolSetReadOnly\(\)](#)  
[DSVariablePoolSetReadWrite\(\)](#)  
[DSVariablePoolLock](#)

**7.29.2.15 void DSVariableRelease (DSVariable \* *aVariable*)**

Function to decrease variable retain count by one.

[DSVariable](#) object is made to decrease its retain count by one, when the retain count hits zero, the function [DSVariableFree\(\)](#) is invoked, freeing the memory of the [DSVariable](#) object. [DSVariable](#) objects do not have an equivalent to autorelease, forcing the developer to invoke a [DSRelease](#) for each [DSRetain](#) explicitly called.

**Parameters**

*aVariable* The variable which will have its retain count reduced.

**See also**

[DSVariableRetain](#)  
[DSVariableFree](#)

**7.29.2.16 DSVariable\* DSVariableRetain (DSVariable \* *aVariable*)**

Function to increase variable retain count by one.

Variables utilize a similar memory management system used in Objective-C NSObject subclasses. A [DSVariable](#) recently allocated begins with a retain count of one.

**Parameters**

*aVariable* The variable which will have its retain count increased.

**Returns**

The same variable which received the retain count increase is returned, for convinience.

**See also**

[DSVariableRelease](#)

## 7.30 DSVariableTokenizer.c File Reference

Implementation file with functions for tokenizing with matrices.

```
#include <stdio.h>
```

```
#include "DSVariableTokenizer.h"
```

Include dependency graph for DSVariableTokenizer.c:

### Functions

- struct [variable\\_token](#) \* **DSVariableTokenAlloc** ()
- void **DSVariableTokenFree** (struct [variable\\_token](#) \*root)
- void **DSVariableTokenSetString** (struct [variable\\_token](#) \*root, char \*string)
- void **DSVariableTokenSetDouble** (struct [variable\\_token](#) \*root, double value)
- char \* **DSVariableTokenString** (struct [variable\\_token](#) \*root)
- double **DSVariableTokenDouble** (struct [variable\\_token](#) \*root)

### 7.30.1 Detailed Description

Implementation file with functions for tokenizing with matrices. Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

#### Author

Jason Lomnitz.

#### Date

2011

## 7.31 DSVariableTokenizerLex.c File Reference

Implementation file with functions for tokenizing matrices, generated by flex.

```
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include "DSTypes.h"
#include "DSMemoryManager.h"
#include "DSVariable.h"
#include "DSVariableTokenizer.h"
#include <unistd.h>
```

Include dependency graph for DSVariableTokenizerLex.c:

### Data Structures

- struct [yy\\_buffer\\_state](#)
- struct [yy\\_trans\\_info](#)
- struct [yyguts\\_t](#)

### Defines

- #define **YY\_INT\_ALIGNED** short int
- #define **FLEX\_SCANNER**
- #define **YY\_FLEX\_MAJOR\_VERSION** 2
- #define **YY\_FLEX\_MINOR\_VERSION** 5
- #define **YY\_FLEX\_SUBMINOR\_VERSION** 35
- #define **INT16\_MIN** (-32767-1)
- #define **INT32\_MIN** (-2147483647-1)
- #define **INT8\_MAX** (127)
- #define **INT16\_MAX** (32767)
- #define **INT32\_MAX** (2147483647)
- #define **UINT8\_MAX** (255U)
- #define **UINT16\_MAX** (65535U)
- #define **UINT32\_MAX** (4294967295U)
- #define **yyconst**
- #define **YY\_NULL** 0
- #define **YY\_SC\_TO\_UI**(c) ((unsigned int) (unsigned char) c)
- #define **YY\_TYPEDEF\_Y Scanner\_T**
- #define **yyin** yyg->yyin\_r
- #define **yyout** yyg->yyout\_r
- #define **yyextra** yyg->yyextra\_r
- #define **yylen** yyg->yylen\_r
- #define **yytext** yyg->yytext\_r
- #define **yylineno** (YY\_CURRENT\_BUFFER\_LVALUE->yy\_bs\_lineno)

---

```

• #define yycolumn (YY_CURRENT_BUFFER_LVALUE->yy_bs_column)
• #define yy_flex_debug yyg->yy_flex_debug_r
• #define BEGIN yyg->yy_start = 1 + 2 *
• #define YY_START ((yyg->yy_start - 1) / 2)
• #define YYSTATE YY_START
• #define YY_STATE_EOF(state) (YY_END_OF_BUFFER + state + 1)
• #define YY_NEW_FILE DSVariableFlexrestart(yyin ,yyscanner )
• #define YY_END_OF_BUFFER_CHAR 0
• #define YY_BUF_SIZE 16384
• #define YY_STATE_BUF_SIZE ((YY_BUF_SIZE + 2) * sizeof(yy_state_type))
• #define YY_TYPEDEF_Y_BUFFER_STATE
• #define YY_TYPEDEF_Y_SIZE_T
• #define EOB_ACT_CONTINUE_SCAN 0
• #define EOB_ACT_END_OF_FILE 1
• #define EOB_ACT_LAST_MATCH 2
• #define YY_LESS_LINENO(n)
• #define yyless(n)
• #define unput(c) yyunput( c, yyg->ytext_ptr , yyscanner )
• #define YY_STRUCT_Y_BUFFER_STATE
• #define YY_BUFFER_NEW 0
• #define YY_BUFFER_NORMAL 1
• #define YY_BUFFER_EOF_PENDING 2
• #define YY_CURRENT_BUFFER
• #define YY_CURRENT_BUFFER_LVALUE yyg->yy_buffer_stack[yyg->yy_buffer_stack_top]

• #define YY_FLUSH_BUFFER DSVariableFlex_flush_buffer(YY_CURRENT_BUFFER ,yyscan-
ner)
• #define yy_new_buffer DSVariableFlex_create_buffer
• #define yy_set_interactive(is_interactive)
• #define yy_set_bol(at_bol)
• #define YY_AT_BOL() (YY_CURRENT_BUFFER_LVALUE->yy_at_bol)
• #define ytext_ptr ytext_r
• #define YY_DO_BEFORE_ACTION
• #define YY_NUM_RULES 14
• #define YY_END_OF_BUFFER 15
• #define REJECT reject_used_but_not_detected
• #define ymore() ymore_used_but_not_detected
• #define YY_MORE_ADJ 0
• #define YY_RESTORE_Y_MORE_OFFSET
• #define malloc(x) DSSecureMalloc(x)
• #define calloc(x, y) DSSecureCalloc(x, y)
• #define realloc(x, y) DSSecureRealloc(x, y)
• #define INITIAL 0
• #define YY_EXTRA_TYPE struct variable\_token *
• #define YY_READ_BUF_SIZE 8192
• #define ECHO fwrite( ytext, yyleng, 1, yyout )
• #define YY_INPUT(buf, result, max_size)
• #define yyterminate() return YY_NULL
• #define YY_START_STACK_INCR 25
• #define YY_FATAL_ERROR(msg) yy_fatal_error( msg , yyscanner)

```

---

- `#define YY_DECL_IS_OURS 1`
- `#define YY_DECL int DSVariableFlexlex (yyscan_t yyscanner)`
- `#define YY_USER_ACTION`
- `#define YY_BREAK break;`
- `#define YY_RULE_SETUP YY_USER_ACTION`
- `#define YY_EXIT_FAILURE 2`
- `#define yyless(n)`
- `#define YYTABLES_NAME "yytables"`

## Typedefs

- typedef signed char `flex_int8_t`
- typedef short int `flex_int16_t`
- typedef int `flex_int32_t`
- typedef unsigned char `flex_uint8_t`
- typedef unsigned short int `flex_uint16_t`
- typedef unsigned int `flex_uint32_t`
- typedef void \* `yyscan_t`
- typedef struct `yy_buffer_state` \* `YY_BUFFER_STATE`
- typedef size\_t `yy_size_t`
- typedef unsigned char `YY_CHAR`
- typedef int `yy_state_type`

## Functions

- void `DSVariableFlexrestart` (FILE \*input\_file, yyscan\_t yyscanner)
- void `DSVariableFlex_switch_to_buffer` (YY\_BUFFER\_STATE new\_buffer, yyscan\_t yyscanner)
- YY\_BUFFER\_STATE `DSVariableFlex_create_buffer` (FILE \*file, int size, yyscan\_t yyscanner)
- void `DSVariableFlex_delete_buffer` (YY\_BUFFER\_STATE b, yyscan\_t yyscanner)
- void `DSVariableFlex_flush_buffer` (YY\_BUFFER\_STATE b, yyscan\_t yyscanner)
- void `DSVariableFlexpush_buffer_state` (YY\_BUFFER\_STATE new\_buffer, yyscan\_t yyscanner)
- void `DSVariableFlexpop_buffer_state` (yyscan\_t yyscanner)
- YY\_BUFFER\_STATE `DSVariableFlex_scan_buffer` (char \*base, yy\_size\_t size, yyscan\_t yyscanner)
- YY\_BUFFER\_STATE `DSVariableFlex_scan_string` (yyconst char \*yy\_str, yyscan\_t yyscanner)
- YY\_BUFFER\_STATE `DSVariableFlex_scan_bytes` (yyconst char \*bytes, yy\_size\_t len, yyscan\_t yyscanner)
- void \* `DSVariableFlexalloc` (yy\_size\_t, yyscan\_t yyscanner)
- void \* `DSVariableFlexrealloc` (void \*, yy\_size\_t, yyscan\_t yyscanner)
- void `DSVariableFlexfree` (void \*, yyscan\_t yyscanner)
- int `DSVariableFlexlex_init` (yyscan\_t \*scanner)
- int `DSVariableFlexlex_init_extra` (YY\_EXTRA\_TYPE user\_defined, yyscan\_t \*scanner)
- int `DSVariableFlexlex_destroy` (yyscan\_t yyscanner)
- int `DSVariableFlexget_debug` (yyscan\_t yyscanner)
- void `DSVariableFlexset_debug` (int debug\_flag, yyscan\_t yyscanner)
- YY\_EXTRA\_TYPE `DSVariableFlexget_extra` (yyscan\_t yyscanner)
- void `DSVariableFlexset_extra` (YY\_EXTRA\_TYPE user\_defined, yyscan\_t yyscanner)
- FILE \* `DSVariableFlexget_in` (yyscan\_t yyscanner)
- void `DSVariableFlexset_in` (FILE \*in\_str, yyscan\_t yyscanner)

- FILE \* **DSVariableFlexget\_out** (yyscan\_t yyscanner)
- void **DSVariableFlexset\_out** (FILE \*out\_str, yyscan\_t yyscanner)
- yy\_size\_t **DSVariableFlexget\_leng** (yyscan\_t yyscanner)
- char \* **DSVariableFlexget\_text** (yyscan\_t yyscanner)
- int **DSVariableFlexget\_lineno** (yyscan\_t yyscanner)
- void **DSVariableFlexset\_lineno** (int line\_number, yyscan\_t yyscanner)
- int **DSVariableFlexwrap** (yyscan\_t yyscanner)
- int **DSVariableFlexlex** (yyscan\_t yyscanner)
- **if** (!yyg->yy\_init)
- **while** (1)
- int **isatty** (int)
- int **DSVariableFlexget\_column** (yyscan\_t yyscanner)
- void **DSVariableFlexset\_column** (int column\_no, yyscan\_t yyscanner)
- struct **variable\_token** \* **DSVariablePoolTokenizeString** (const char \*string)

## Variables

- YY\_DECL register yy\_state\_type **yy\_current\_state**
- register char \* **yy\_cp**
- register char \* **yy\_bp**
- register int **yy\_act**
- struct **yyguts\_t** \* **yyg** = (struct **yyguts\_t**\*)yyscanner

### 7.31.1 Detailed Description

Implementation file with functions for tokenizing matrices, generated by flex. This file was generated directly by the flex program, and is the source code responsible for matrix tokenization. This file was generated by flex, according to a specification written by Jason Lomnitz. To generate this file, the following command must be executed: "flex -t DSVariableGrammar.l > DSVariableTokenizerLex.c".

Copyright (C) 2011 Jason Lomnitz.

This file is part of the Design Space Toolbox V2 (C Library).

The Design Space Toolbox V2 is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

The Design Space Toolbox V2 is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with the Design Space Toolbox. If not, see <<http://www.gnu.org/licenses/>>.

## Author

Jason Lomnitz.

## Date

2011



## 7.31.2 Define Documentation

### 7.31.2.1 #define YY\_CURRENT\_BUFFER

**Value:**

```
( yyg->yy_buffer_stack \
    ? yyg->yy_buffer_stack[yyg->yy_buffer_stack_top] \
    : NULL)
```

### 7.31.2.2 #define YY\_DO\_BEFORE\_ACTION

**Value:**

```
yyg->yytext_ptr = yy_bp; \
    yylen = (size_t) (yy_cp - yy_bp); \
    yyg->yy_hold_char = *yy_cp; \
    *yy_cp = '\0'; \
    yyg->yy_c_buf_p = yy_cp;
```

### 7.31.2.3 #define YY\_INPUT(buf, result, max\_size)

**Value:**

```
if ( YY_CURRENT_BUFFER_LVALUE->yy_is_interactive ) \
    { \
        int c = '*'; \
        yy_size_t n; \
        for ( n = 0; n < max_size && \
              (c = getc( yyin )) != EOF && c != '\n'; ++n ) \
            buf[n] = (char) c; \
        if ( c == '\n' ) \
            buf[n++] = (char) c; \
        if ( c == EOF && ferror( yyin ) ) \
            YY_FATAL_ERROR( "input in flex scanner failed" ); \
        result = n; \
    } \
else \
    { \
        errno=0; \
        while ( (result = fread(buf, 1, max_size, yyin))==0 && ferror(yyi \
n)) \
        { \
            if( errno != EINTR) \
            { \
                YY_FATAL_ERROR( "input in flex scanner failed" ); \
                break; \
            } \
            errno=0; \
            clearerr(yyin); \
        } \
    } \
\
```

### 7.31.2.4 #define yy\_set\_bol(at\_bol)

**Value:**

```
{ \
    if ( ! YY_CURRENT_BUFFER ){ \
        DSVariableFlexensure_buffer_stack (yyscanner); \
        YY_CURRENT_BUFFER_LVALUE = \
            DSVariableFlex_create_buffer (yyin,YY_BUF_SIZE ,yyscanner); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_at_bol = at_bol; \
}
```

### 7.31.2.5 #define yy\_set\_interactive(is\_interactive)

**Value:**

```
{ \
    if ( ! YY_CURRENT_BUFFER ){ \
        DSVariableFlexensure_buffer_stack (yyscanner); \
        YY_CURRENT_BUFFER_LVALUE = \
            DSVariableFlex_create_buffer (yyin,YY_BUF_SIZE ,yyscanner); \
    } \
    YY_CURRENT_BUFFER_LVALUE->yy_is_interactive = is_interactive; \
}
```

### 7.31.2.6 #define yyless(n)

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO (yyless_macro_arg); \
        yytext[yy leng] = yyg->yy_hold_char; \
        yyg->yy_c_buf_p = yytext + yyless_macro_arg; \
        yyg->yy_hold_char = *yyg->yy_c_buf_p; \
        *yyg->yy_c_buf_p = '\0'; \
        yy leng = yyless_macro_arg; \
    } \
while ( 0 )
```

### 7.31.2.7 #define yyless(n)

**Value:**

```
do \
    { \
        /* Undo effects of setting up yytext. */ \
        int yyless_macro_arg = (n); \
        YY_LESS_LINENO (yyless_macro_arg); \
        *yy_cp = yyg->yy_hold_char; \
        YY_RESTORE_YY_MORE_OFFSET \
        yyg->yy_c_buf_p = yy_cp = yy_bp + yyless_macro_arg - YY_MORE_ADJ; \
    } \
    YY_DO_BEFORE_ACTION; /* set up yytext again */ \
    } \
while ( 0 )
```

### 7.31.3 Function Documentation

#### 7.31.3.1 YY\_BUFFER\_STATE DSVariableFlex\_create\_buffer (FILE \**file*, int *size*, yyscan\_t *yyscanner*)

Allocate and initialize an input buffer state.

##### Parameters

*file* A readable stream.

*size* The character buffer size in bytes. When in doubt, use YY\_BUF\_SIZE.

*yyscanner* The scanner object.

##### Returns

the allocated buffer state.

#### 7.31.3.2 void DSVariableFlex\_delete\_buffer (YY\_BUFFER\_STATE *b*, yyscan\_t *yyscanner*)

Destroy the buffer.

##### Parameters

*b* a buffer created with [DSVariableFlex\\_create\\_buffer\(\)](#)

*yyscanner* The scanner object.

#### 7.31.3.3 void DSVariableFlex\_flush\_buffer (YY\_BUFFER\_STATE *b*, yyscan\_t *yyscanner*)

Discard all buffered characters. On the next scan, YY\_INPUT will be called.

##### Parameters

*b* the buffer state to be flushed, usually YY\_CURRENT\_BUFFER.

*yyscanner* The scanner object.

#### 7.31.3.4 YY\_BUFFER\_STATE DSVariableFlex\_scan\_buffer (char \**base*, yy\_size\_t *size*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan directly from a user-specified character buffer.

##### Parameters

*base* the character buffer

*size* the size in bytes of the character buffer

*yyscanner* The scanner object.

##### Returns

the newly allocated buffer state object.

### 7.31.3.5 YY\_BUFFER\_STATE DSVariableFlex\_scan\_bytes (yyconst char \* *yybytes*, yy\_size\_t *\_yybytes\_len*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan the given bytes. The next call to DSVariableFlexlex() will scan from a *copy* of *bytes*.

#### Parameters

*bytes* the byte buffer to scan  
*len* the number of bytes in the buffer pointed to by *bytes*.  
*yyscanner* The scanner object.

#### Returns

the newly allocated buffer state object.

### 7.31.3.6 YY\_BUFFER\_STATE DSVariableFlex\_scan\_string (yyconst char \* *yyst*, yyscan\_t *yyscanner*)

Setup the input buffer state to scan a string. The next call to DSVariableFlexlex() will scan from a *copy* of *str*.

#### Parameters

*yyst* a NUL-terminated string to scan  
*yyscanner* The scanner object.

#### Returns

the newly allocated buffer state object.

#### Note

If you want to scan bytes that may contain NUL values, then use [DSVariableFlex\\_scan\\_bytes\(\)](#) instead.

### 7.31.3.7 void DSVariableFlex\_switch\_to\_buffer (YY\_BUFFER\_STATE *new\_buffer*, yyscan\_t *yyscanner*)

Switch to a different input buffer.

#### Parameters

*new\_buffer* The new input buffer.  
*yyscanner* The scanner object.

### 7.31.3.8 int DSVariableFlexget\_column (yyscan\_t *yyscanner*)

Get the current column number.

#### Parameters

*yyscanner* The scanner object.

**7.31.3.9 YY\_EXTRA\_TYPE DSVariableFlexget\_extra (yyscan\_t *yyscanner*)**

Get the user-defined data for this scanner.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.10 FILE \* DSVariableFlexget\_in (yyscan\_t *yyscanner*)**

Get the input stream.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.11 yy\_size\_t DSVariableFlexget\_leng (yyscan\_t *yyscanner*)**

Get the length of the current token.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.12 int DSVariableFlexget\_lineno (yyscan\_t *yyscanner*)**

Get the current line number.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.13 FILE \* DSVariableFlexget\_out (yyscan\_t *yyscanner*)**

Get the output stream.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.14 char \* DSVariableFlexget\_text (yyscan\_t *yyscanner*)**

Get the current token.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.15 void DSVariableFlexpop\_buffer\_state (yyscan\_t *yyscanner*)**

Removes and deletes the top of the stack, if present. The next element becomes the new top.

**Parameters**

*yyscanner* The scanner object.

**7.31.3.16 void DSVariableFlexpush\_buffer\_state (YY\_BUFFER\_STATE *new\_buffer*, *yyscan\_t* *yyscanner*)**

Pushes the new state onto the stack. The new state becomes the current state. This function will allocate the stack if necessary.

**Parameters**

*new\_buffer* The new state.

*yyscanner* The scanner object.

**7.31.3.17 void DSVariableFlexrestart (FILE \* *input\_file*, *yyscan\_t* *yyscanner*)**

Immediately switch to a different input stream.

**Parameters**

*input\_file* A readable stream.

*yyscanner* The scanner object.

**Note**

This function does not reset the start condition to `INITIAL`.

**7.31.3.18 void DSVariableFlexset\_column (int *column\_no*, *yyscan\_t* *yyscanner*)**

Set the current column.

**Parameters**

*line\_number*

*yyscanner* The scanner object.

**7.31.3.19 void DSVariableFlexset\_extra (YY\_EXTRA\_TYPE *user\_defined*, *yyscan\_t* *yyscanner*)**

Set the user-defined data. This data is never touched by the scanner.

**Parameters**

*user\_defined* The data to be associated with this scanner.

*yyscanner* The scanner object.

**7.31.3.20 void DSVariableFlexset\_in (FILE \* *in\_str*, yyscan\_t *yyscanner*)**

Set the input stream. This does not discard the current input buffer.

**Parameters**

*in\_str* A readable stream.

*yyscanner* The scanner object.

**See also**

[DSVariableFlex\\_switch\\_to\\_buffer](#)

**7.31.3.21 void DSVariableFlexset\_lineno (int *line\_number*, yyscan\_t *yyscanner*)**

Set the current line number.

**Parameters**

*line\_number*

*yyscanner* The scanner object.

**7.31.4 Variable Documentation****7.31.4.1 YY\_DECL register yy\_state\_type yy\_current\_state**

The main scanner function which does all the work.

# Index

- [\\_varDictionary, 19](#)
- [Actions for DS Errors., 12](#)
- [DS\\_DICTONARY\\_ACCESSORY](#)
  - [dsInternalDictionarySetValue, 9](#)
  - [dsInternalDictionaryValue, 9](#)
- [DS\\_IO\\_TAG\\_TYPES, 14](#)
- [ds\\_parallelstack\\_t, 22](#)
- [DS\\_VARIABLE\\_ACCESSORY](#)
  - [DSVariableName, 18](#)
  - [DSVariableSetValue, 18](#)
  - [DSVariableValue, 18](#)
- [DSCase, 23](#)
- [DSCasePrintingOptions](#)
  - [DSIO.c, 91](#)
- [DSCaseStringInJSONFormat](#)
  - [DSIO.c, 88](#)
  - [DSIO.h, 94](#)
- [DSComplexMatrix](#)
  - [DSTypes.h, 148](#)
- [DSDesignSpace, 25](#)
- [DSDesignSpace.c, 51](#)
- [DSDesignSpace.h, 54](#)
- [DSDesignSpaceParallel.c, 56](#)
- [DSDesignSpaceParallel.h, 58](#)
- [DSDictionary, 26](#)
- [DSError](#)
  - [DSErrors.h, 65](#)
- [DSErrorFunction](#)
  - [DSErrors.c, 62](#)
  - [DSErrors.h, 65](#)
- [DSErrors.c, 60](#)
  - [DSErrorFunction, 62](#)
  - [MSIZE, 61](#)
  - [STACK\\_TRACE\\_NUM, 61](#)
- [DSErrors.h, 63](#)
  - [DSError, 65](#)
  - [DSErrorFunction, 65](#)
- [DSExpression](#)
  - [DSTypes.h, 148](#)
- [dsexpression, 27](#)
- [DSExpression.c, 66](#)
- [DSExpression.h, 68](#)
- [DSExpressionFlex\\_flush\\_buffer](#)
  - [DSExpressionTokenizerLex.c, 75](#)
- [DSExpressionFlex\\_scan\\_buffer](#)
  - [DSExpressionTokenizerLex.c, 75](#)
- [DSExpressionFlex\\_scan\\_bytes](#)
  - [DSExpressionTokenizerLex.c, 76](#)
- [DSExpressionFlex\\_scan\\_string](#)
  - [DSExpressionTokenizerLex.c, 76](#)
- [DSExpressionFlexget\\_column](#)
  - [DSExpressionTokenizerLex.c, 76](#)
- [DSExpressionFlexget\\_extra](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexget\\_in](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexget\\_leng](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexget\\_lineno](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexget\\_out](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexget\\_text](#)
  - [DSExpressionTokenizerLex.c, 77](#)
- [DSExpressionFlexpop\\_buffer\\_state](#)
  - [DSExpressionTokenizerLex.c, 78](#)
- [DSExpressionFlexpush\\_buffer\\_state](#)
  - [DSExpressionTokenizerLex.c, 78](#)
- [DSExpressionFlexset\\_column](#)
  - [DSExpressionTokenizerLex.c, 78](#)
- [DSExpressionFlexset\\_extra](#)
  - [DSExpressionTokenizerLex.c, 78](#)
- [DSExpressionFlexset\\_in](#)
  - [DSExpressionTokenizerLex.c, 78](#)
- [DSExpressionFlexset\\_lineno](#)
  - [DSExpressionTokenizerLex.c, 79](#)
- [DSExpressionTokenizerLex.c, 70](#)
  - [DSExpressionFlex\\_flush\\_buffer, 75](#)
  - [DSExpressionFlex\\_scan\\_buffer, 75](#)
  - [DSExpressionFlex\\_scan\\_bytes, 76](#)
  - [DSExpressionFlex\\_scan\\_string, 76](#)
  - [DSExpressionFlexget\\_column, 76](#)
  - [DSExpressionFlexget\\_extra, 77](#)
  - [DSExpressionFlexget\\_in, 77](#)
  - [DSExpressionFlexget\\_leng, 77](#)
  - [DSExpressionFlexget\\_lineno, 77](#)
  - [DSExpressionFlexget\\_out, 77](#)
  - [DSExpressionFlexget\\_text, 77](#)



- DSExpressionFlexpop\_buffer\_state, 78
- DSExpressionFlexpush\_buffer\_state, 78
- DSExpressionFlexset\_column, 78
- DSExpressionFlexset\_extra, 78
- DSExpressionFlexset\_in, 78
- DSExpressionFlexset\_lineno, 79
- YY\_CURRENT\_BUFFER, 73
- YY\_DO\_BEFORE\_ACTION, 73
- YY\_INPUT, 74
- yy\_set\_bol, 74
- yy\_set\_interactive, 74
- yylless, 75
- DSGMAACCESSORS, 13
- DSGMASystem, 28
- DSGMASystem.c, 80
- DSGMASystem.h, 82
- DSGMASystemParsingAux.h, 84
  - gma\_parseraux\_t, 85
- DSInternalDictionary
  - DSTypes.h, 148
- dsInternalDictionarySetValue
  - DS\_DICTIONARY\_ACCESSORY, 9
- dsInternalDictionaryValue
  - DS\_DICTIONARY\_ACCESSORY, 9
- DSIO.c, 86
  - DSCasePrintingOptions, 91
  - DSCaseStringInJSONFormat, 88
  - DSIOSetCaseJSONOptions, 88
  - DSIOSetErrorFile, 88
  - DSIOSetPostErrorFunction, 88
  - DSIOSetPostFatalErrorFunction, 89
  - DSIOSetPostWarningFunction, 89
  - DSIOSetPrintFunction, 89
  - DSIOSetSSystemJSONOptions, 89
  - DSMatrixArrayStringInJSONFormat, 90
  - DSMatrixStringInJSONFormat, 90
  - DSSSystemPrintingOptions, 91
  - DSSSystemStringInJSONFormat, 90
  - DSVariablePoolStringInJSONFormat, 91
- DSIO.h, 92
  - DSCaseStringInJSONFormat, 94
  - DSIOErrorFile, 98
  - DSIOSetCaseJSONOptions, 94
  - DSIOSetErrorFile, 95
  - DSIOSetPostErrorFunction, 95
  - DSIOSetPostFatalErrorFunction, 95
  - DSIOSetPostWarningFunction, 95
  - DSIOSetPrintFunction, 96
  - DSIOSetSSystemJSONOptions, 96
  - DSMatrixArrayStringInJSONFormat, 96
  - DSMatrixStringInJSONFormat, 97
  - DSPostError, 98
  - DSPostFatalError, 98
  - DSPostWarning, 98
  - DSPrintf, 99
  - DSSSystemStringInJSONFormat, 97
  - DSVariablePoolStringInJSONFormat, 97
- DSIOErrorFile
  - DSIO.h, 98
- DSIOSetCaseJSONOptions
  - DSIO.c, 88
  - DSIO.h, 94
- DSIOSetErrorFile
  - DSIO.c, 88
  - DSIO.h, 95
- DSIOSetPostErrorFunction
  - DSIO.c, 88
  - DSIO.h, 95
- DSIOSetPostFatalErrorFunction
  - DSIO.c, 89
  - DSIO.h, 95
- DSIOSetPostWarningFunction
  - DSIO.c, 89
  - DSIO.h, 95
- DSIOSetPrintFunction
  - DSIO.c, 89
  - DSIO.h, 96
- DSIOSetSSystemJSONOptions
  - DSIO.c, 89
  - DSIO.h, 96
- DSLCKLocked
  - DSTypes.h, 149
- DSLCKReadOnly
  - DSTypes.h, 149
- DSLCKReadWrite
  - DSTypes.h, 149
- DSLCKReadWriteAdd
  - DSTypes.h, 149
- DSMatrix, 29
- DSMatrix.h, 100
  - DSMatrixAlloc, 103
  - DSMatrixByAddingMatrix, 103
  - DSMatrixByParsingString, 103
  - DSMatrixBySubtractingMatrix, 104
  - DSMatrixCalloc, 104
  - DSMatrixCopy, 104
  - DSMatrixDoubleValue, 105
  - DSMatrixFree, 105
  - DSMatrixIdentity, 105
  - DSMatrixPLUDecomposition, 106
  - DSMatrixRandomNumbers, 106
  - DSMatrixSetDoubleValueAll, 106
- DSMatrix\_gsl.c, 107
  - DSMatrixAlloc, 110
  - DSMatrixByAddingMatrix, 110
  - DSMatrixByParsingString, 111
  - DSMatrixBySubtractingMatrix, 111
  - DSMatrixCalloc, 111

- DSMatrixCopy, [112](#)
- DSMatrixDoubleValue, [112](#)
- DSMatrixFree, [112](#)
- DSMatrixIdentity, [113](#)
- DSMatrixPLUDecomposition, [113](#)
- DSMatrixRandomNumbers, [113](#)
- DSMatrixSetDoubleValueAll, [113](#)
- DSMatrixAlloc
  - DSMatrix.h, [103](#)
  - DSMatrix\_gsl.c, [110](#)
- DSMatrixArray, [30](#)
- DSMatrixArray.c, [115](#)
  - DSMatrixArrayAddMatrix, [116](#)
  - DSMatrixArrayAlloc, [116](#)
  - DSMatrixArrayCopy, [116](#)
  - DSMatrixArrayFree, [117](#)
  - DSMatrixArrayMatrix, [117](#)
- DSMatrixArray.h, [118](#)
  - DSMatrixArrayAddMatrix, [119](#)
  - DSMatrixArrayAlloc, [119](#)
  - DSMatrixArrayCopy, [120](#)
  - DSMatrixArrayFree, [120](#)
  - DSMatrixArrayMatrix, [120](#)
- DSMatrixArrayAddMatrix
  - DSMatrixArray.c, [116](#)
  - DSMatrixArray.h, [119](#)
- DSMatrixArrayAlloc
  - DSMatrixArray.c, [116](#)
  - DSMatrixArray.h, [119](#)
- DSMatrixArrayCopy
  - DSMatrixArray.c, [116](#)
  - DSMatrixArray.h, [120](#)
- DSMatrixArrayFree
  - DSMatrixArray.c, [117](#)
  - DSMatrixArray.h, [120](#)
- DSMatrixArrayMatrix
  - DSMatrixArray.c, [117](#)
  - DSMatrixArray.h, [120](#)
- DSMatrixArrayStringInJSONFormat
  - DSIO.c, [90](#)
  - DSIO.h, [96](#)
- DSMatrixByAddingMatrix
  - DSMatrix.h, [103](#)
  - DSMatrix\_gsl.c, [110](#)
- DSMatrixByParsingString
  - DSMatrix.h, [103](#)
  - DSMatrix\_gsl.c, [111](#)
- DSMatrixBySubtractingMatrix
  - DSMatrix.h, [104](#)
  - DSMatrix\_gsl.c, [111](#)
- DSMatrixCalloc
  - DSMatrix.h, [104](#)
  - DSMatrix\_gsl.c, [111](#)
- DSMatrixCopy
  - DSMatrix.h, [104](#)
  - DSMatrix\_gsl.c, [112](#)
- DSMatrixDoubleValue
  - DSMatrix.h, [105](#)
  - DSMatrix\_gsl.c, [112](#)
- DSMatrixFlex\_flush\_buffer
  - DSMatrixTokenizerLex.c, [129](#)
- DSMatrixFlex\_scan\_buffer
  - DSMatrixTokenizerLex.c, [129](#)
- DSMatrixFlex\_scan\_bytes
  - DSMatrixTokenizerLex.c, [130](#)
- DSMatrixFlex\_scan\_string
  - DSMatrixTokenizerLex.c, [130](#)
- DSMatrixFlexget\_column
  - DSMatrixTokenizerLex.c, [130](#)
- DSMatrixFlexget\_extra
  - DSMatrixTokenizerLex.c, [130](#)
- DSMatrixFlexget\_in
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexget\_leng
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexget\_lineno
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexget\_out
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexget\_text
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexpop\_buffer\_state
  - DSMatrixTokenizerLex.c, [131](#)
- DSMatrixFlexpush\_buffer\_state
  - DSMatrixTokenizerLex.c, [132](#)
- DSMatrixFlexset\_column
  - DSMatrixTokenizerLex.c, [132](#)
- DSMatrixFlexset\_extra
  - DSMatrixTokenizerLex.c, [132](#)
- DSMatrixFlexset\_in
  - DSMatrixTokenizerLex.c, [132](#)
- DSMatrixFlexset\_lineno
  - DSMatrixTokenizerLex.c, [132](#)
- DSMatrixFree
  - DSMatrix.h, [105](#)
  - DSMatrix\_gsl.c, [112](#)
- DSMatrixIdentity
  - DSMatrix.h, [105](#)
  - DSMatrix\_gsl.c, [113](#)
- DSMatrixPLUDecomposition
  - DSMatrix.h, [106](#)
  - DSMatrix\_gsl.c, [113](#)
- DSMatrixRandomNumbers
  - DSMatrix.h, [106](#)
  - DSMatrix\_gsl.c, [113](#)
- DSMatrixSetDoubleValueAll
  - DSMatrix.h, [106](#)
  - DSMatrix\_gsl.c, [113](#)

- DSMatrixStringInJSONFormat
  - DSIO.c, [90](#)
  - DSIO.h, [97](#)
- DSMatrixTokenizer.c, [121](#)
- DSMatrixTokenizer.h, [122](#)
- DSMatrixTokenizerLex.c, [124](#)
  - DSMatrixFlex\_flush\_buffer, [129](#)
  - DSMatrixFlex\_scan\_buffer, [129](#)
  - DSMatrixFlex\_scan\_bytes, [130](#)
  - DSMatrixFlex\_scan\_string, [130](#)
  - DSMatrixFlexget\_column, [130](#)
  - DSMatrixFlexget\_extra, [130](#)
  - DSMatrixFlexget\_in, [131](#)
  - DSMatrixFlexget\_leng, [131](#)
  - DSMatrixFlexget\_lineno, [131](#)
  - DSMatrixFlexget\_out, [131](#)
  - DSMatrixFlexget\_text, [131](#)
  - DSMatrixFlexpop\_buffer\_state, [131](#)
  - DSMatrixFlexpush\_buffer\_state, [132](#)
  - DSMatrixFlexset\_column, [132](#)
  - DSMatrixFlexset\_extra, [132](#)
  - DSMatrixFlexset\_in, [132](#)
  - DSMatrixFlexset\_lineno, [132](#)
  - YY\_CURRENT\_BUFFER, [127](#)
  - YY\_DO\_BEFORE\_ACTION, [127](#)
  - YY\_INPUT, [128](#)
  - yy\_set\_bol, [128](#)
  - yy\_set\_interactive, [128](#)
  - yyless, [129](#)
- DSMemoryManager.c, [134](#)
  - DSSecureCalloc, [135](#)
  - DSSecureFree, [135](#)
  - DSSecureMalloc, [135](#)
  - DSSecureRealloc, [136](#)
- DSMemoryManager.h, [137](#)
  - DSSecureCalloc, [138](#)
  - DSSecureFree, [138](#)
  - DSSecureMalloc, [138](#)
  - DSSecureRealloc, [139](#)
- DSPostError
  - DSIO.h, [98](#)
- DSPostFatalError
  - DSIO.h, [98](#)
- DSPostWarning
  - DSIO.h, [98](#)
- DSPrintf
  - DSIO.h, [99](#)
- DSSecureCalloc
  - DSMemoryManager.c, [135](#)
  - DSMemoryManager.h, [138](#)
- DSSecureFree
  - DSMemoryManager.c, [135](#)
  - DSMemoryManager.h, [138](#)
- DSSecureMalloc
  - DSMemoryManager.c, [135](#)
  - DSMemoryManager.h, [138](#)
- DSMemoryManager.c, [135](#)
- DSMemoryManager.h, [138](#)
- DSSecureRealloc, [136](#)
- DSSysACCESSORS, [17](#)
- DSSSystem, [31](#)
- DSSSystem.h, [140](#)
- DSSystemPrintingOptions
  - DSIO.c, [91](#)
- DSSystemStringInJSONFormat
  - DSIO.c, [90](#)
  - DSIO.h, [97](#)
- DSSStack, [32](#)
- DSStd.h, [142](#)
- DSSymbolicMatrix, [33](#)
- DSSymbolicMatrix.h, [144](#)
- DSTypes.h, [146](#)
  - DSComplexMatrix, [148](#)
  - DSExpression, [148](#)
  - DSInternalDictionary, [148](#)
  - DSLockLocked, [149](#)
  - DSLockReadOnly, [149](#)
  - DSLockReadWrite, [149](#)
  - DSLockReadWriteAdd, [149](#)
  - DSVariablePoolLock, [149](#)
- DSVariable, [34](#)
- DSVariable.c, [150](#)
  - DSVariableAlloc, [152](#)
  - DSVariableFree, [152](#)
  - DSVariablePoolAddVariable, [153](#)
  - DSVariablePoolAddVariableWithName, [153](#)
  - DSVariablePoolAlloc, [153](#)
  - DSVariablePoolCopy, [153](#)
  - DSVariablePoolFree, [154](#)
  - DSVariablePoolIsReadOnly, [154](#)
  - DSVariablePoolIsReadWrite, [154](#)
  - DSVariablePoolIsReadWriteAdd, [155](#)
  - DSVariablePoolNumberOfVariables, [155](#)
  - DSVariablePoolSetReadOnly, [155](#)
  - DSVariablePoolSetReadWrite, [155](#)
  - DSVariablePoolSetReadWriteAdd, [156](#)
  - DSVariableRelease, [156](#)
  - DSVariableRetain, [156](#)
- DSVariable.h, [158](#)
  - DSVariableAlloc, [160](#)
  - DSVariableFree, [160](#)
  - DSVariablePoolAddVariable, [161](#)
  - DSVariablePoolAddVariableWithName, [161](#)
  - DSVariablePoolAlloc, [161](#)
  - DSVariablePoolCopy, [161](#)
  - DSVariablePoolFree, [162](#)
  - DSVariablePoolIsReadOnly, [162](#)
  - DSVariablePoolIsReadWrite, [162](#)

- DSVariablePoolIsReadWriteAdd, 163
- DSVariablePoolNumberOfVariables, 163
- DSVariablePoolSetReadOnly, 163
- DSVariablePoolSetReadWrite, 163
- DSVariablePoolSetReadWriteAdd, 164
- DSVariableRelease, 164
- DSVariableRetain, 164
- DSVariableAlloc
  - DSVariable.c, 152
  - DSVariable.h, 160
- DSVariableFlex\_create\_buffer
  - DSVariableTokenizerLex.c, 173
- DSVariableFlex\_delete\_buffer
  - DSVariableTokenizerLex.c, 173
- DSVariableFlex\_flush\_buffer
  - DSVariableTokenizerLex.c, 173
- DSVariableFlex\_scan\_buffer
  - DSVariableTokenizerLex.c, 173
- DSVariableFlex\_scan\_bytes
  - DSVariableTokenizerLex.c, 173
- DSVariableFlex\_scan\_string
  - DSVariableTokenizerLex.c, 174
- DSVariableFlex\_switch\_to\_buffer
  - DSVariableTokenizerLex.c, 174
- DSVariableFlexget\_column
  - DSVariableTokenizerLex.c, 174
- DSVariableFlexget\_extra
  - DSVariableTokenizerLex.c, 174
- DSVariableFlexget\_in
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexget\_leng
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexget\_lineno
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexget\_out
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexget\_text
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexpop\_buffer\_state
  - DSVariableTokenizerLex.c, 175
- DSVariableFlexpush\_buffer\_state
  - DSVariableTokenizerLex.c, 176
- DSVariableFlexrestart
  - DSVariableTokenizerLex.c, 176
- DSVariableFlexset\_column
  - DSVariableTokenizerLex.c, 176
- DSVariableFlexset\_extra
  - DSVariableTokenizerLex.c, 176
- DSVariableFlexset\_in
  - DSVariableTokenizerLex.c, 176
- DSVariableFlexset\_lineno
  - DSVariableTokenizerLex.c, 177
- DSVariableFree
  - DSVariable.c, 152
- DSVariable.h, 160
- DSVariableName
  - DS\_VARIABLE\_ACCESSORY, 18
- DSVariablePool, 35
- DSVariablePoolAddVariable
  - DSVariable.c, 153
  - DSVariable.h, 161
- DSVariablePoolAddVariableWithName
  - DSVariable.c, 153
  - DSVariable.h, 161
- DSVariablePoolAlloc
  - DSVariable.c, 153
  - DSVariable.h, 161
- DSVariablePoolCopy
  - DSVariable.c, 153
  - DSVariable.h, 161
- DSVariablePoolFree
  - DSVariable.c, 154
  - DSVariable.h, 162
- DSVariablePoolIsReadOnly
  - DSVariable.c, 154
  - DSVariable.h, 162
- DSVariablePoolIsReadWrite
  - DSVariable.c, 154
  - DSVariable.h, 162
- DSVariablePoolIsReadWriteAdd
  - DSVariable.c, 155
  - DSVariable.h, 163
- DSVariablePoolLock
  - DSTypes.h, 149
- DSVariablePoolNumberOfVariables
  - DSVariable.c, 155
  - DSVariable.h, 163
- DSVariablePoolSetReadOnly
  - DSVariable.c, 155
  - DSVariable.h, 163
- DSVariablePoolSetReadWrite
  - DSVariable.c, 155
  - DSVariable.h, 163
- DSVariablePoolSetReadWriteAdd
  - DSVariable.c, 156
  - DSVariable.h, 164
- DSVariablePoolStringInJSONFormat
  - DSIO.c, 91
  - DSIO.h, 97
- DSVariableRelease
  - DSVariable.c, 156
  - DSVariable.h, 164
- DSVariableRetain
  - DSVariable.c, 156
  - DSVariable.h, 164
- DSVariableSetValue
  - DS\_VARIABLE\_ACCESSORY, 18
- DSVariableTokenizer.c, 166

- DSVariableTokenizerLex.c, 167
  - DSVariableFlex\_create\_buffer, 173
  - DSVariableFlex\_delete\_buffer, 173
  - DSVariableFlex\_flush\_buffer, 173
  - DSVariableFlex\_scan\_buffer, 173
  - DSVariableFlex\_scan\_bytes, 173
  - DSVariableFlex\_scan\_string, 174
  - DSVariableFlex\_switch\_to\_buffer, 174
  - DSVariableFlexget\_column, 174
  - DSVariableFlexget\_extra, 174
  - DSVariableFlexget\_in, 175
  - DSVariableFlexget\_leng, 175
  - DSVariableFlexget\_lineno, 175
  - DSVariableFlexget\_out, 175
  - DSVariableFlexget\_text, 175
  - DSVariableFlexpop\_buffer\_state, 175
  - DSVariableFlexpush\_buffer\_state, 176
  - DSVariableFlexrestart, 176
  - DSVariableFlexset\_column, 176
  - DSVariableFlexset\_extra, 176
  - DSVariableFlexset\_in, 176
  - DSVariableFlexset\_lineno, 177
  - YY\_CURRENT\_BUFFER, 171
  - yy\_current\_state, 177
  - YY\_DO\_BEFORE\_ACTION, 171
  - YY\_INPUT, 171
  - yy\_set\_bol, 171
  - yy\_set\_interactive, 172
  - yyless, 172
- DSVariableValue
  - DS\_VARIABLE\_ACCESSORY, 18
- DSVertices, 36
- expression\_token, 37
- gma\_parseraux\_t
  - DSGMASystemParsingAux.h, 85
- Macros to manipulate dictionary nodes., 9
- Macros to manipulate variables., 18
- matrix\_token, 38
- Messages for DS Errors., 10
- MSIZE
  - DSErrors.c, 61
- Options for JSON conversion of DSCase object., 15
- Options for JSON conversion of DSSSystem object., 16
- parse\_expression\_s, 39
- parser\_aux, 40
- parser\_aux::base\_info, 21
- pthread\_struct, 41
- STACK\_TRACE\_NUM
  - DSErrors.c, 61
- v\_token\_data, 42
- variable\_token, 43
- yy\_bs\_column
  - yy\_buffer\_state, 44
- yy\_bs\_lineno
  - yy\_buffer\_state, 44
- yy\_buffer\_stack
  - yyguts\_t, 46
- yy\_buffer\_stack\_max
  - yyguts\_t, 46
- yy\_buffer\_stack\_top
  - yyguts\_t, 46
- yy\_buffer\_state, 44
  - yy\_bs\_column, 44
  - yy\_bs\_lineno, 44
- YY\_CURRENT\_BUFFER
  - DSEExpressionTokenizerLex.c, 73
  - DSMatrixTokenizerLex.c, 127
  - DSVariableTokenizerLex.c, 171
- yy\_current\_state
  - DSVariableTokenizerLex.c, 177
- YY\_DO\_BEFORE\_ACTION
  - DSEExpressionTokenizerLex.c, 73
  - DSMatrixTokenizerLex.c, 127
  - DSVariableTokenizerLex.c, 171
- YY\_INPUT
  - DSEExpressionTokenizerLex.c, 74
  - DSMatrixTokenizerLex.c, 128
  - DSVariableTokenizerLex.c, 171
- yy\_set\_bol
  - DSEExpressionTokenizerLex.c, 74
  - DSMatrixTokenizerLex.c, 128
  - DSVariableTokenizerLex.c, 171
- yy\_set\_interactive
  - DSEExpressionTokenizerLex.c, 74
  - DSMatrixTokenizerLex.c, 128
  - DSVariableTokenizerLex.c, 172
- yy\_trans\_info, 45
- yyguts\_t, 46
  - yy\_buffer\_stack, 46
  - yy\_buffer\_stack\_max, 46
  - yy\_buffer\_stack\_top, 46
- yyless
  - DSEExpressionTokenizerLex.c, 75
  - DSMatrixTokenizerLex.c, 129
  - DSVariableTokenizerLex.c, 172
- YYMINORTYPE, 47
- yyParser, 48
- yyStackEntry, 49