

openCMISS
0.1

Generated by Doxygen 1.5.7.1

Wed Oct 22 16:05:26 2008

Contents

1	openCMISS Documentation	1
1.1	LICENSE	1
2	Obtaining the Code and Setting up the Development Environment	3
2.1	Obtaining the Code and Libraries	4
2.1.1	Obtain the Code	4
2.1.2	Obtain the Libraries	4
2.1.3	Makefile Structure	4
2.2	Programmer documentation	5
2.3	Project Setup	5
2.3.1	On AIX 5.3 (HPC)	5
2.3.1.1	Set environment	5
2.3.1.2	Set MPI	5
2.3.1.3	Compile	5
2.3.2	On Ubuntu 8.04	5
2.3.2.1	Set environment	5
2.3.2.2	Install Compilers	6
2.3.2.3	Compile	6
2.3.3	On Windows XP (Visual Studio 2005)	6
2.3.3.1	Install Compilers	6
2.3.3.2	Install MPI	7
2.3.3.3	Compile and Debug	7
2.3.3.4	Run	7
2.3.4	On Windows Vista (Visual Studio 2008)	7
2.3.4.1	7
2.3.4.2	Compile	7
2.4	Libraries Build (Optional)	8
2.4.1	Compiling PETSc	8

2.4.1.1	Step1: Linux Environment installation and Compiler Environment Set up (For Windows only)	8
2.4.1.2	Step2: Compile PETSC	8
3	Simple Test	11
4	Todo List	13
5	Module Documentation	19
5.1	BASE_ROUTINES::OutputType	19
5.1.1	Detailed Description	19
5.1.2	Variable Documentation	19
5.1.2.1	DIAGNOSTIC_OUTPUT_TYPE	19
5.1.2.2	ERROR_OUTPUT_TYPE	20
5.1.2.3	GENERAL_OUTPUT_TYPE	20
5.1.2.4	HELP_OUTPUT_TYPE	20
5.1.2.5	TIMING_OUTPUT_TYPE	21
5.2	BASE_ROUTINES::FileUnits	22
5.2.1	Detailed Description	23
5.2.2	Variable Documentation	23
5.2.2.1	DIAGNOSTICS_FILE_UNIT	23
5.2.2.2	ECHO_FILE_UNIT	23
5.2.2.3	IO1_FILE_UNIT	23
5.2.2.4	IO2_FILE_UNIT	23
5.2.2.5	IO3_FILE_UNIT	24
5.2.2.6	IO4_FILE_UNIT	24
5.2.2.7	IO5_FILE_UNIT	24
5.2.2.8	LEARN_FILE_UNIT	24
5.2.2.9	OPEN_COMFILE_UNIT	24
5.2.2.10	START_READ_COMFILE_UNIT	25
5.2.2.11	STOP_READ_COMFILE_UNIT	25
5.2.2.12	TEMPORARY_FILE_UNIT	25
5.2.2.13	TIMING_FILE_UNIT	25
5.3	BASE_ROUTINES::DiagnosticTypes	26
5.3.1	Detailed Description	26
5.3.2	Variable Documentation	26
5.3.2.1	ALL_DIAG_TYPE	26
5.3.2.2	FROM_DIAG_TYPE	26

5.3.2.3	IN_DIAG_TYPE	27
5.4	BASE_ROUTINES::TimingTypes	28
5.4.1	Detailed Description	28
5.4.2	Variable Documentation	28
5.4.2.1	ALL_TIMING_TYPE	28
5.4.2.2	FROM_TIMING_TYPE	28
5.4.2.3	IN_TIMING_TYPE	29
5.5	COORDINATE_ROUTINES::CoordinateSystemTypes	30
5.5.1	Detailed Description	30
5.5.2	Variable Documentation	30
5.5.2.1	COORDINATE_CYLINDRICAL_POLAR_TYPE	30
5.5.2.2	COORDINATE_OBLATE_SPHEROIDAL_TYPE	31
5.5.2.3	COORDINATE_PROLATE_SPHEROIDAL_TYPE	31
5.5.2.4	COORDINATE_RECTANGULAR_CARTESIAN_TYPE	32
5.5.2.5	COORDINATE_SPHERICAL_POLAR_TYPE	32
5.6	COORDINATE_ROUTINES::RadialInterpolations	34
5.6.1	Detailed Description	34
5.6.2	Variable Documentation	34
5.6.2.1	COORDINATE_NO_RADIAL_INTERPOLATION_TYPE	34
5.6.2.2	COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE	35
5.6.2.3	COORDINATE_RADIAL_INTERPOLATION_TYPE	35
5.6.2.4	COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE	35
5.7	COORDINATE_ROUTINES::JacobianType	36
5.7.1	Detailed Description	36
5.7.2	Variable Documentation	36
5.7.2.1	COORDINATE_JACOBIAN_AREA_TYPE	36
5.7.2.2	COORDINATE_JACOBIAN_LINE_TYPE	36
5.7.2.3	COORDINATE_JACOBIAN_VOLUME_TYPE	37
5.8	DOMAIN_MAPPINGS::DomainType	38
5.8.1	Detailed Description	38
5.8.2	Variable Documentation	38
5.8.2.1	DOMAIN_LOCAL_BOUNDARY	38
5.8.2.2	DOMAIN_LOCAL_GHOST	38
5.8.2.3	DOMAIN_LOCAL_INTERNAL	39
5.9	EQUATIONS_SET_CONSTANTS::SetupTypes	40
5.9.1	Detailed Description	41

5.9.2	Variable Documentation	41
5.9.2.1	EQUATIONS_SET_SETUP_ANALYTIC_TYPE	41
5.9.2.2	EQUATIONS_SET_SETUP_DEPENDENT_TYPE	41
5.9.2.3	EQUATIONS_SET_SETUP_EQUATIONS_TYPE	41
5.9.2.4	EQUATIONS_SET_SETUP_FINAL_TYPE	42
5.9.2.5	EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE	42
5.9.2.6	EQUATIONS_SET_SETUP_GEOMETRY_TYPE	42
5.9.2.7	EQUATIONS_SET_SETUP_INITIAL_TYPE	42
5.9.2.8	EQUATIONS_SET_SETUP_MATERIALS_TYPE	43
5.9.2.9	EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE	43
5.9.2.10	EQUATIONS_SET_SETUP_SOURCE_TYPE	43
5.10	EQUATIONS_SET_CONSTANTS::SetupActionTypes	44
5.10.1	Detailed Description	44
5.10.2	Variable Documentation	44
5.10.2.1	EQUATIONS_SET_SETUP_FINISH_ACTION	44
5.10.2.2	EQUATIONS_SET_SETUP_START_ACTION	44
5.11	EQUATIONS_SET_CONSTANTS::FixedConditions	46
5.11.1	Detailed Description	46
5.11.2	Variable Documentation	46
5.11.2.1	EQUATIONS_SET_FIXED_BOUNDARY_CONDITION	46
5.11.2.2	EQUATIONS_SET_MIXED_BOUNDARY_CONDITION	46
5.11.2.3	EQUATIONS_SET_NOT_FIXED	47
5.12	EQUATIONS_SET_CONSTANTS::LinearityTypes	48
5.12.1	Detailed Description	48
5.12.2	Variable Documentation	48
5.12.2.1	EQUATIONS_SET_LINEAR	48
5.12.2.2	EQUATIONS_SET_NONLINEAR	48
5.12.2.3	EQUATIONS_SET_NONLINEAR_BCS	49
5.12.2.4	NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES	49
5.13	EQUATIONS_SET_CONSTANTS::TimeDepedenceTypes	50
5.13.1	Detailed Description	50
5.13.2	Variable Documentation	50
5.13.2.1	EQUATIONS_SET_DYNAMIC	50
5.13.2.2	EQUATIONS_SET_QUASISTATIC	50
5.13.2.3	EQUATIONS_SET_STATIC	51
5.13.2.4	NUMBER_OF_EQUATIONS_SET_TIME_TYPES	51

5.14 EQUATIONS_SET_CONSTANTS::SolutionMethods	52
5.14.1 Detailed Description	52
5.14.2 Variable Documentation	52
5.14.2.1 EQUATIONS_SET_BEM SOLUTION_METHOD	52
5.14.2.2 EQUATIONS_SET_FD SOLUTION_METHOD	53
5.14.2.3 EQUATIONS_SET_FEM SOLUTION_METHOD	53
5.14.2.4 EQUATIONS_SET_FV SOLUTION_METHOD	53
5.14.2.5 EQUATIONS_SET_GFEM SOLUTION_METHOD	53
5.14.2.6 EQUATIONS_SET_GFV SOLUTION_METHOD	54
5.14.2.7 NUMBER_OF_EQUATIONS_SET SOLUTION_METHODS	54
5.15 EQUATIONS_SET_CONSTANTS::OutputTypes	55
5.15.1 Detailed Description	55
5.15.2 Variable Documentation	55
5.15.2.1 EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT	55
5.15.2.2 EQUATIONS_SET_MATRIX_OUTPUT	56
5.15.2.3 EQUATIONS_SET_NO_OUTPUT	56
5.15.2.4 EQUATIONS_SET_TIMING_OUTPUT	56
5.16 EQUATIONS_SET_CONSTANTS::SparsityTypes	57
5.16.1 Detailed Description	57
5.16.2 Variable Documentation	57
5.16.2.1 EQUATIONS_SET_FULL_MATRICES	57
5.16.2.2 EQUATIONS_SET_SPARSE_MATRICES	57
5.17 FIELD_ROUTINES::DependentTypes	58
5.17.1 Detailed Description	58
5.17.2 Variable Documentation	58
5.17.2.1 FIELD_DEPENDENT_TYPE	58
5.17.2.2 FIELD_INDEPENDENT_TYPE	58
5.18 FIELD_ROUTINES::DimensionTypes	59
5.18.1 Detailed Description	59
5.18.2 Variable Documentation	59
5.18.2.1 FIELD_SCALAR_DIMENSION_TYPE	59
5.18.2.2 FIELD_VECTOR_DIMENSION_TYPE	59
5.19 FIELD_ROUTINES::FieldTypes	60
5.19.1 Detailed Description	60
5.19.2 Variable Documentation	60
5.19.2.1 FIELD_FIBRE_TYPE	60

5.19.2.2	FIELD_GENERAL_TYPE	60
5.19.2.3	FIELD_GEOMETRIC_TYPE	61
5.19.2.4	FIELD_MATERIAL_TYPE	61
5.20	FIELD_ROUTINES::InterpolationTypes	62
5.20.1	Detailed Description	62
5.20.2	Variable Documentation	62
5.20.2.1	FIELD_CONSTANT_INTERPOLATION	62
5.20.2.2	FIELD_ELEMENT_BASED_INTERPOLATION	63
5.20.2.3	FIELD_NODE_BASED_INTERPOLATION	63
5.20.2.4	FIELD_POINT_BASED_INTERPOLATION	63
5.21	FIELD_ROUTINES::VariableTypes	64
5.21.1	Detailed Description	64
5.21.2	Variable Documentation	64
5.21.2.1	FIELD_NORMAL_VARIABLE_TYPE	64
5.21.2.2	FIELD_NUMBER_OF_VARIABLE_TYPES	65
5.21.2.3	FIELD_STANDARD_VARIABLE_TYPE	65
5.21.2.4	FIELD_TIME_DERIV1_VARIABLE_TYPE	65
5.21.2.5	FIELD_TIME_DERIV2_VARIABLE_TYPE	66
5.22	FIELD_ROUTINES::DofTypes	67
5.22.1	Detailed Description	67
5.22.2	Variable Documentation	67
5.22.2.1	FIELD_CONSTANT_DOF_TYPE	67
5.22.2.2	FIELD_ELEMENT_DOF_TYPE	67
5.22.2.3	FIELD_NODE_DOF_TYPE	68
5.22.2.4	FIELD_POINT_DOF_TYPE	68
5.23	FIELD_ROUTINES::ParameterSetTypes	69
5.23.1	Detailed Description	69
5.23.2	Variable Documentation	69
5.23.2.1	FIELD_BOUNDARY_CONDITIONS_SET_TYPE	69
5.23.2.2	FIELD_INITIAL_CONDITIONS_SET_TYPE	70
5.23.2.3	FIELD_NUMBER_OF_SET_TYPES	70
5.23.2.4	FIELD_VALUES_SET_TYPE	70
5.24	FIELD_ROUTINES::ScalingTypes	71
5.24.1	Detailed Description	71
5.24.2	Variable Documentation	71
5.24.2.1	FIELD_ARC_LENGTH_SCALING	71

5.24.2.2	FIELD_ARITHMETIC_MEAN_SCALING	72
5.24.2.3	FIELD_HARMONIC_MEAN_SCALING	72
5.24.2.4	FIELD_NO_SCALING	72
5.24.2.5	FIELD_UNIT_SCALING	72
5.25	GENERATED_MESH_ROUTINES::GeneratedMeshTypes	73
5.25.1	Detailed Description	73
5.25.2	Variable Documentation	73
5.25.2.1	GENERATED_MESH_FRACTAL_TREE_MESH_TYPE	73
5.25.2.2	GENERATED_MESH_POLAR_MESH_TYPE	73
5.25.2.3	GENERATED_MESH_REGULAR_MESH_TYPE	74
5.26	INPUT_OUTPUT::MatrixNameIndexFormat	75
5.26.1	Detailed Description	75
5.26.2	Variable Documentation	75
5.26.2.1	WRITE_STRING_MATRIX_NAME_AND_INDICES	75
5.26.2.2	WRITE_STRING_MATRIX_NAME_ONLY	75
5.27	KINDS::IntegerKinds	77
5.27.1	Detailed Description	77
5.27.2	Variable Documentation	77
5.27.2.1	INTG	77
5.27.2.2	LINTG	77
5.27.2.3	PTR	78
5.27.2.4	SINTG	79
5.28	KINDS::RealKinds	80
5.28.1	Detailed Description	80
5.28.2	Variable Documentation	80
5.28.2.1	DP	80
5.28.2.2	QP	80
5.28.2.3	SP	81
5.29	KINDS::ComplexKinds	82
5.29.1	Detailed Description	82
5.29.2	Variable Documentation	82
5.29.2.1	_DP	82
5.29.2.2	_SP	83
5.29.2.3	DPC	83
5.29.2.4	SPC	83
5.30	LISTS::DataType	84

5.30.1	Detailed Description	84
5.30.2	Variable Documentation	84
5.30.2.1	LIST_DP_TYPE	84
5.30.2.2	LIST_INTG_TYPE	84
5.30.2.3	LIST_SP_TYPE	85
5.31	LISTS::SortingOrder	86
5.31.1	Detailed Description	86
5.31.2	Variable Documentation	86
5.31.2.1	LIST_SORT_ASCENDING_TYPE	86
5.31.2.2	LIST_SORT_DESCENDING_TYPE	86
5.31.2.3	LIST_UNSORTED_TYPE	87
5.32	LISTS::SortingMethod	88
5.32.1	Detailed Description	88
5.32.2	Variable Documentation	88
5.32.2.1	LIST_BUBBLE_SORT_METHOD	88
5.32.2.2	LIST_HEAP_SORT_METHOD	88
5.32.2.3	LIST_SHELL_SORT_METHOD	89
5.33	MATRIX_VECTOR::DataTypes	90
5.33.1	Detailed Description	90
5.33.2	Variable Documentation	90
5.33.2.1	MATRIX_VECTOR_DP_TYPE	90
5.33.2.2	MATRIX_VECTOR_INTG_TYPE	91
5.33.2.3	MATRIX_VECTOR_L_TYPE	91
5.33.2.4	MATRIX_VECTOR_SP_TYPE	92
5.34	MATRIX_VECTOR::StorageTypes	93
5.34.1	Detailed Description	93
5.34.2	Variable Documentation	93
5.34.2.1	MATRIX_BLOCK_STORAGE_TYPE	93
5.34.2.2	MATRIX_COLUMN_MAJOR_STORAGE_TYPE	94
5.34.2.3	MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE	94
5.34.2.4	MATRIX_COMPRESSED_ROW_STORAGE_TYPE	95
5.34.2.5	MATRIX_DIAGONAL_STORAGE_TYPE	95
5.34.2.6	MATRIX_ROW_COLUMN_STORAGE_TYPE	95
5.34.2.7	MATRIX_ROW_MAJOR_STORAGE_TYPE	96
5.35	MESH_ROUTINES::DecompositionTypes	97
5.35.1	Detailed Description	97

5.35.2	Variable Documentation	97
5.35.2.1	DECOMPOSITION_ALL_TYPE	97
5.35.2.2	DECOMPOSITION_CALCULATED_TYPE	97
5.35.2.3	DECOMPOSITION_USER_DEFINED_TYPE	98
5.36	PROBLEM_CONSTANTS::SetupTypes	99
5.36.1	Detailed Description	99
5.36.2	Variable Documentation	99
5.36.2.1	PROBLEM_SETUP_CONTROL_TYPE	99
5.36.2.2	PROBLEM_SETUP_INITIAL_TYPE	100
5.36.2.3	PROBLEM_SETUP SOLUTION_TYPE	100
5.36.2.4	PROBLEM_SETUP_SOLVER_TYPE	100
5.37	PROBLEM_CONSTANTS::SetupActionTypes	101
5.37.1	Detailed Description	101
5.37.2	Variable Documentation	101
5.37.2.1	PROBLEM_SETUP_DO_ACTION	101
5.37.2.2	PROBLEM_SETUP_FINISH_ACTION	101
5.37.2.3	PROBLEM_SETUP_START_ACTION	102
5.38	PROBLEM_CONSTANTS::SolutionOutputTypes	103
5.38.1	Detailed Description	103
5.38.2	Variable Documentation	103
5.38.2.1	PROBLEM SOLUTION MATRIX_OUTPUT	103
5.38.2.2	PROBLEM SOLUTION NO_OUTPUT	103
5.38.2.3	PROBLEM SOLUTION TIMING_OUTPUT	104
5.39	PROBLEM_CONSTANTS::SolverOutputTypes	105
5.39.1	Detailed Description	105
5.39.2	Variable Documentation	105
5.39.2.1	PROBLEM SOLVER_NO_OUTPUT	105
5.39.2.2	PROBLEM SOLVER_SOLVER_OUTPUT	105
5.39.2.3	PROBLEM SOLVER_TIMING_OUTPUT	106
5.40	PROBLEM_CONSTANTS::SolverSparsityTypes	107
5.40.1	Detailed Description	107
5.40.2	Variable Documentation	107
5.40.2.1	PROBLEM SOLVER_FULL_MATRICES	107
5.40.2.2	PROBLEM SOLVER_SPARSE_MATRICES	107
5.41	PROBLEM_ROUTINES::LinearityTypes	108
5.41.1	Detailed Description	108

5.41.2	Variable Documentation	108
5.41.2.1	NUMBER_OF_PROBLEM_LINEARITIES	108
5.41.2.2	PROBLEM_LINEAR	108
5.41.2.3	PROBLEM_NONLINEAR	109
5.41.2.4	PROBLEM_NONLINEAR_BCS	109
5.42	PROBLEM_ROUTINES::TimeDepedenceTypes	110
5.42.1	Detailed Description	110
5.42.2	Variable Documentation	110
5.42.2.1	NUMBER_OF_PROBLEM_TIME_TYPES	110
5.42.2.2	PROBLEM_DYNAMIC	110
5.42.2.3	PROBLEM_QUASISTATIC	111
5.42.2.4	PROBLEM_STATIC	111
5.43	SOLVER_ROUTINES::SolverTypes	112
5.43.1	Detailed Description	112
5.43.2	Variable Documentation	112
5.43.2.1	SOLVER_EIGENPROBLEM_TYPE	112
5.43.2.2	SOLVER_LINEAR_TYPE	112
5.43.2.3	SOLVER_NONLINEAR_TYPE	113
5.43.2.4	SOLVER_TIME_INTEGRATION_TYPE	113
5.44	SOLVER_ROUTINES::SolverLibraries	114
5.44.1	Detailed Description	114
5.44.2	Variable Documentation	114
5.44.2.1	SOLVER_CMISS_LIBRARY	114
5.44.2.2	SOLVER_PETSC_LIBRARY	114
5.45	SOLVER_ROUTINES::LinearSolverTypes	116
5.45.1	Detailed Description	116
5.45.2	Variable Documentation	116
5.45.2.1	SOLVER_LINEAR_DIRECT_SOLVE_TYPE	116
5.45.2.2	SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE	116
5.46	SOLVER_ROUTINES::DirectLinearSolverTypes	118
5.46.1	Detailed Description	118
5.46.2	Variable Documentation	118
5.46.2.1	SOLVER_DIRECT_CHOLESKY	118
5.46.2.2	SOLVER_DIRECT_LU	118
5.46.2.3	SOLVER_DIRECT_SVD	119
5.47	SOLVER_ROUTINES::IterativeLinearSolverTypes	120

5.47.1	Detailed Description	120
5.47.2	Variable Documentation	120
5.47.2.1	SOLVER_ITERATIVE_BICGSTAB	120
5.47.2.2	SOLVER_ITERATIVE_BICONJUGATE_GRADIENT	121
5.47.2.3	SOLVER_ITERATIVE_CHEBYCHEV	121
5.47.2.4	SOLVER_ITERATIVE_CONJGRAD_SQUARED	121
5.47.2.5	SOLVER_ITERATIVE_CONJUGATE_GRADIENT	121
5.47.2.6	SOLVER_ITERATIVE_GMRES	122
5.47.2.7	SOLVER_ITERATIVE_RICHARDSON	122
5.48	SOLVER_ROUTINES::IterativePreconditionerTypes	123
5.48.1	Detailed Description	123
5.48.2	Variable Documentation	123
5.48.2.1	SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER	123
5.48.2.2	SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER	124
5.48.2.3	SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_- PRECONDITIONER	124
5.48.2.4	SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER	124
5.48.2.5	SOLVER_ITERATIVE_JACOBI_PRECONDITIONER	125
5.48.2.6	SOLVER_ITERATIVE_NO_PRECONDITIONER	125
5.48.2.7	SOLVER_ITERATIVE_SOR_PRECONDITIONER	125
5.49	SOLVER_ROUTINES::OutputTypes	126
5.49.1	Detailed Description	126
5.49.2	Variable Documentation	126
5.49.2.1	SOLVER_MATRIX_OUTPUT	126
5.49.2.2	SOLVER_NO_OUTPUT	126
5.49.2.3	SOLVER_SOLVER_OUTPUT	127
5.49.2.4	SOLVER_TIMING_OUTPUT	127
5.50	SOLVER_ROUTINES::SparsityTypes	128
5.50.1	Detailed Description	128
5.50.2	Variable Documentation	128
5.50.2.1	SOLVER_FULL_MATRICES	128
5.50.2.2	SOLVER_SPARSE_MATRICES	128
5.51	TREES::TreeNodeColourTypes	129
5.51.1	Detailed Description	129
5.51.2	Variable Documentation	129
5.51.2.1	TREE_BLACK_NODE	129
5.51.2.2	TREE_RED_NODE	129

5.52 TREES::TreeNodeInsertStatus	130
5.52.1 Detailed Description	130
5.52.2 Variable Documentation	130
5.52.2.1 TREE_NODE_DUPLICATE_KEY	130
5.52.2.2 TREE_NODE_INSERT_SUCESSFUL	130
5.53 TREES::TreeInsertTypes	131
5.53.1 Detailed Description	131
5.53.2 Variable Documentation	131
5.53.2.1 TREE_DUPLICATES_ALLOWED_TYPE	131
5.53.2.2 TREE_NO_DUPLICATES_ALLOWED	131
6 Namespace Documentation	133
6.1 BASE_ROUTINES Namespace Reference	133
6.1.1 Detailed Description	138
6.1.2 Function Documentation	138
6.1.2.1 BASE_ROUTINES_FINALISE	138
6.1.2.2 BASE_ROUTINES_INITIALISE	138
6.1.2.3 DIAGNOSTICS_SET_OFF	138
6.1.2.4 DIAGNOSTICS_SET_ON	139
6.1.2.5 ENTERS	139
6.1.2.6 ERRORS	148
6.1.2.7 EXITS	158
6.1.2.8 FLAG_ERROR_C	167
6.1.2.9 FLAG_ERROR_VS	167
6.1.2.10 FLAG_WARNING_C	168
6.1.2.11 FLAG_WARNING_VS	168
6.1.2.12 OUTPUT_SET_OFF	168
6.1.2.13 OUTPUT_SET_ON	168
6.1.2.14 TIMING_SET_OFF	169
6.1.2.15 TIMING_SET_ON	169
6.1.2.16 TIMING_SUMMARY_OUTPUT	169
6.1.2.17 WRITE_STR	170
6.1.3 Variable Documentation	171
6.1.3.1 DIAG_ALL_SUBROUTINES	171
6.1.3.2 DIAG_FILE_OPEN	171
6.1.3.3 DIAG_FROM_SUBROUTINE	171
6.1.3.4 DIAG_OR_TIMING	171

6.1.3.5	DIAG_ROUTINE_LIST	171
6.1.3.6	DIAGNOSTICS	171
6.1.3.7	DIAGNOSTICS1	171
6.1.3.8	DIAGNOSTICS2	172
6.1.3.9	DIAGNOSTICS3	172
6.1.3.10	DIAGNOSTICS4	172
6.1.3.11	DIAGNOSTICS5	172
6.1.3.12	DIAGNOSTICS_LEVEL1	172
6.1.3.13	DIAGNOSTICS_LEVEL2	172
6.1.3.14	DIAGNOSTICS_LEVEL3	173
6.1.3.15	DIAGNOSTICS_LEVEL4	173
6.1.3.16	DIAGNOSTICS_LEVEL5	173
6.1.3.17	ECHO_OUTPUT	173
6.1.3.18	MAX_OUTPUT_LINES	173
6.1.3.19	OP_STRING	173
6.1.3.20	ROUTINE_STACK	174
6.1.3.21	TIMING	174
6.1.3.22	TIMING_ALL_SUBROUTINES	174
6.1.3.23	TIMING_FILE_OPEN	174
6.1.3.24	TIMING_FROM_SUBROUTINE	175
6.1.3.25	TIMING_ROUTINE_LIST	175
6.1.3.26	TIMING_SUMMARY	175
6.2	BINARY_FILE Namespace Reference	176
6.2.1	Detailed Description	177
6.2.2	Function Documentation	177
6.2.2.1	CLOSE_BINARY_FILE	177
6.2.2.2	CLOSE_CMISS_BINARY_FILE	178
6.2.2.3	INQUIRE_EOF_BINARY_FILE	178
6.2.2.4	INQUIRE_OPEN_BINARY_FILE	178
6.2.2.5	OPEN_BINARY_FILE	178
6.2.2.6	OPEN_CMISS_BINARY_FILE	178
6.2.2.7	READ_BINARY_FILE_CHARACTER	179
6.2.2.8	READ_BINARY_FILE_DP	179
6.2.2.9	READ_BINARY_FILE_DP1	179
6.2.2.10	READ_BINARY_FILE_DPC	179
6.2.2.11	READ_BINARY_FILE_DPC1	180

6.2.2.12	READ_BINARY_FILE_INTG	180
6.2.2.13	READ_BINARY_FILE_INTG1	180
6.2.2.14	READ_BINARY_FILE_LINTG	180
6.2.2.15	READ_BINARY_FILE_LINTG1	181
6.2.2.16	READ_BINARY_FILE_LOGICAL	181
6.2.2.17	READ_BINARY_FILE_LOGICAL1	181
6.2.2.18	READ_BINARY_FILE_SINTG	181
6.2.2.19	READ_BINARY_FILE_SINTG1	181
6.2.2.20	READ_BINARY_FILE_SP	182
6.2.2.21	READ_BINARY_FILE_SP1	182
6.2.2.22	READ_BINARY_FILE_SPC	182
6.2.2.23	READ_BINARY_FILE_SPC1	182
6.2.2.24	READ_BINARY_TAG_HEADER	183
6.2.2.25	RESET_BINARY_NUMBER_TAGS	183
6.2.2.26	SET_BINARY_FILE	183
6.2.2.27	SKIP_BINARY_FILE	183
6.2.2.28	SKIP_BINARY_TAGS	183
6.2.2.29	SKIP_CM_BINARY_HEADER	184
6.2.2.30	WRITE_BINARY_FILE_CHARACTER	184
6.2.2.31	WRITE_BINARY_FILE_DP	184
6.2.2.32	WRITE_BINARY_FILE_DP1	184
6.2.2.33	WRITE_BINARY_FILE_DPC	184
6.2.2.34	WRITE_BINARY_FILE_DPC1	185
6.2.2.35	WRITE_BINARY_FILE_INTG	185
6.2.2.36	WRITE_BINARY_FILE_INTG1	185
6.2.2.37	WRITE_BINARY_FILE_LINTG	185
6.2.2.38	WRITE_BINARY_FILE_LINTG1	186
6.2.2.39	WRITE_BINARY_FILE_LOGICAL	186
6.2.2.40	WRITE_BINARY_FILE_LOGICAL1	186
6.2.2.41	WRITE_BINARY_FILE_SINTG	186
6.2.2.42	WRITE_BINARY_FILE_SINTG1	186
6.2.2.43	WRITE_BINARY_FILE_SP	187
6.2.2.44	WRITE_BINARY_FILE_SP1	187
6.2.2.45	WRITE_BINARY_FILE_SPC	187
6.2.2.46	WRITE_BINARY_FILE_SPC1	187
6.2.2.47	WRITE_BINARY_TAG_HEADER	188

6.2.3	Variable Documentation	188
6.2.3.1	BINARY_FILE_READABLE	188
6.2.3.2	BINARY_FILE_USED	188
6.2.3.3	BINARY_FILE_WRITABLE	188
6.2.3.4	CMISS_BINARY_FILE_HEADER	188
6.2.3.5	CMISS_BINARY_HISTORY_FILE	188
6.2.3.6	CMISS_BINARY_IDENTITY	188
6.2.3.7	CMISS_BINARY_IDENTITY_HEADER	188
6.2.3.8	CMISS_BINARY_MACHINE_HEADER	188
6.2.3.9	CMISS_BINARY_MATRIX_FILE	189
6.2.3.10	CMISS_BINARY_SIGNAL_FILE	189
6.2.3.11	FILE_BEGINNING	189
6.2.3.12	FILE_CHANGE_ENDIAN	189
6.2.3.13	FILE_CURRENT	189
6.2.3.14	FILE_END	189
6.2.3.15	FILE SAME_ENDIAN	189
6.2.3.16	MAX_NUM_BINARY_FILES	189
6.3	BLAS Namespace Reference	190
6.3.1	Detailed Description	190
6.4	CLASSICAL_FIELD_ROUTINES Namespace Reference	191
6.4.1	Detailed Description	191
6.4.2	Function Documentation	191
6.4.2.1	CL	191
6.4.2.2	CLASSICAL_FIELD_EQUATIONS_SETFINITEELEMENTCALCULATE	192
6.4.2.3	CLASSICAL_FIELD_EQUATIONS_SETSETUP	193
6.4.2.4	CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET	193
6.4.2.5	CLASSICAL_FIELD_PROBLEM_SETUP	194
6.5	CMISS Namespace Reference	195
6.5.1	Detailed Description	195
6.5.2	Function Documentation	195
6.5.2.1	CMISS_FINALISE	195
6.5.2.2	CMISS_INITIALISE	196
6.5.2.3	CMISS_WRITE_ERROR	196
6.5.3	Variable Documentation	196
6.5.3.1	CMISS_BUILD_VERSION	196
6.5.3.2	CMISS_MAJOR_VERSION	196

6.5.3.3	CMISS_MINOR_VERSION	196
6.6	CMISS_MPI Namespace Reference	197
6.6.1	Detailed Description	197
6.6.2	Function Documentation	197
6.6.2.1	MPI_ERROR_CHECK	197
6.7	CMISS_PARMETIS Namespace Reference	198
6.7.1	Detailed Description	198
6.7.2	Function Documentation	198
6.7.2.1	PARMETIS_PARTKWAY	198
6.7.2.2	PARMETIS_PARTMESHKWAY	199
6.8	CMISS_PETSC Namespace Reference	200
6.8.1	Detailed Description	205
6.8.2	Function Documentation	205
6.8.2.1	PETSC_ERRORHANDLING_SET_OFF	205
6.8.2.2	PETSC_ERRORHANDLING_SET_ON	206
6.8.2.3	PETSC_FINALIZE	206
6.8.2.4	PETSC_INITIALIZE	206
6.8.2.5	PETSC_ISDESTROY	207
6.8.2.6	PETSC_ISFINALISE	207
6.8.2.7	PETSC_ISINITIALISE	207
6.8.2.8	PETSC_ISLOCALTOGLOBALMAPPINGAPPLY	208
6.8.2.9	PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS	208
6.8.2.10	PETSC_ISLOCALTOGLOBALMAPPINGCREATE	208
6.8.2.11	PETSC_ISLOCALTOGLOBALMAPPINGDESTROY	209
6.8.2.12	PETSC_ISLOCALTOGLOBALMAPPINGFINALISE	209
6.8.2.13	PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE	210
6.8.2.14	PETSC_KSPCREATE	210
6.8.2.15	PETSC_KSPDESTROY	210
6.8.2.16	PETSC_KSPFINALISE	211
6.8.2.17	PETSC_KSPGETCONVERGEDREASON	211
6.8.2.18	PETSC_KSPGETITERATIONNUMBER	211
6.8.2.19	PETSC_KSPGETPC	212
6.8.2.20	PETSC_KSPGETRESIDUALNORM	212
6.8.2.21	PETSC_KSPINITIALISE	213
6.8.2.22	PETSC_KSPSETFROMOPTIONS	213
6.8.2.23	PETSC_KSPSETOPERATORS	214

6.8.2.24 PETSC_KSPSETTOLERANCES	214
6.8.2.25 PETSC_KSPSETTYPE	215
6.8.2.26 PETSC_KSPSETUP	215
6.8.2.27 PETSC_KSPSOLVE	215
6.8.2.28 PETSC_LOGPRINTSUMMARY	216
6.8.2.29 PETSC_MATASSEMBLYBEGIN	216
6.8.2.30 PETSC_MATASSEMBLYEND	216
6.8.2.31 PETSC_MATCREATE	217
6.8.2.32 PETSC_MATCREATEMPIAIJ	217
6.8.2.33 PETSC_MATCREATEMPIIDENSE	218
6.8.2.34 PETSC_MATCREATESEQAIJ	218
6.8.2.35 PETSC_MATCREATESEQDENSE	219
6.8.2.36 PETSC_MATDESTROY	219
6.8.2.37 PETSC_MATFINALISE	220
6.8.2.38 PETSC_MATGETARRAY	220
6.8.2.39 PETSC_MATGETOWNERSHIPRANGE	220
6.8.2.40 PETSC_MATGETVALUES	221
6.8.2.41 PETSC_MATINITIALISE	221
6.8.2.42 PETSC_MATRESTOREARRAY	222
6.8.2.43 PETSC_MATSETLOCALTOGLOBALMAPPING	222
6.8.2.44 PETSC_MATSETOPTION	222
6.8.2.45 PETSC_MATSETSIZES	223
6.8.2.46 PETSC_MATSETVALUE	223
6.8.2.47 PETSC_MATSETVALUELOCAL	224
6.8.2.48 PETSC_MATSETVALUES	224
6.8.2.49 PETSC_MATSETVALUESLOCAL	225
6.8.2.50 PETSC_MATVIEW	225
6.8.2.51 PETSC_MATZEROENTRIES	225
6.8.2.52 PETSC_PCFINALISE	226
6.8.2.53 PETSC_PCINITIALISE	226
6.8.2.54 PETSC_PCSETTYPE	227
6.8.2.55 PETSC_VECASSEMBLYBEGIN	227
6.8.2.56 PETSC_VECASSEMBLYEND	227
6.8.2.57 PETSC_VECCREATE	228
6.8.2.58 PETSC_VECCREATEGHOST	228
6.8.2.59 PETSC_VECCREATEGHOSTWITHARRAY	229

6.8.2.60	PETSC_VECCREATEMPI	229
6.8.2.61	PETSC_VECCREATEMPIWITHARRAY	230
6.8.2.62	PETSC_VECCREATESEQ	230
6.8.2.63	PETSC_VECCREATESEQWITHARRAY	230
6.8.2.64	PETSC_VECDESTROY	231
6.8.2.65	PETSC_VECDUPLICATE	231
6.8.2.66	PETSC_VECFINALISE	232
6.8.2.67	PETSC_VECGETARRAY	232
6.8.2.68	PETSC_VECGETARRAYF90	232
6.8.2.69	PETSC_VECGETLOCALSIZE	233
6.8.2.70	PETSC_VECGETOWNERSHIPRANGE	233
6.8.2.71	PETSC_VECGETSIZE	233
6.8.2.72	PETSC_VECGETVALUES	234
6.8.2.73	PETSC_VECGHOSTGETLOCALFORM	234
6.8.2.74	PETSC_VECGHOSTRESTORELOCALFORM	235
6.8.2.75	PETSC_VECGHOSTUPDATEBEGIN	235
6.8.2.76	PETSC_VECGHOSTUPDATEEND	235
6.8.2.77	PETSC_VECINITIALISE	236
6.8.2.78	PETSC_VECRESTOREARRAY	236
6.8.2.79	PETSC_VECRESTOREARRAYF90	236
6.8.2.80	PETSC_VECSET	237
6.8.2.81	PETSC_VECSETFROMOPTIONS	237
6.8.2.82	PETSC_VECSETLOCALTOGLOBALMAPPING	237
6.8.2.83	PETSC_VECSETSIZES	238
6.8.2.84	PETSC_VECSETVALUES	238
6.8.2.85	PETSC_VECSETVALUESLOCAL	239
6.8.2.86	PETSC_VECVIEW	239
6.8.3	Variable Documentation	239
6.8.3.1	PETSC_HANDLE_ERROR	239
6.9	CMISS_PETSC_TYPES Namespace Reference	240
6.9.1	Detailed Description	240
6.10	COMP_ENVIRONMENT Namespace Reference	241
6.10.1	Detailed Description	242
6.10.2	Function Documentation	242
6.10.2.1	COMPUTATIONAL_ENVIRONMENT_FINALISE	242
6.10.2.2	COMPUTATIONAL_ENVIRONMENT_INITIALISE	242

6.10.2.3 COMPUTATIONAL_NODE_FINALISE	243
6.10.2.4 COMPUTATIONAL_NODE_INITIALISE	243
6.10.2.5 COMPUTATIONAL_NODE_MPI_TYPE_FINALISE	244
6.10.2.6 COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE	244
6.10.2.7 COMPUTATIONAL_NODE_NUMBER_GET	245
6.10.2.8 COMPUTATIONAL_NODES_NUMBER_GET	245
6.10.3 Variable Documentation	246
6.10.3.1 COMPUTATIONAL_ENVIRONMENT	246
6.10.3.2 MPI_COMPUTATIONAL_NODE_TYPE_DATA	246
6.11 COORDINATE_ROUTINES Namespace Reference	247
6.11.1 Detailed Description	249
6.11.2 Function Documentation	249
6.11.2.1 CO	249
6.11.2.2 CO	250
6.11.2.3 COORDINATE_CONVERT_FROM_RC_DP	250
6.11.2.4 COORDINATE_CONVERT_FROM_RC_SP	250
6.11.2.5 COORDINATE_CONVERT_TO_RC_DP	251
6.11.2.6 COORDINATE_CONVERT_TO_RC_SP	251
6.11.2.7 COORDINATE_DELTA_CALCULATE_DP	251
6.11.2.8 COORDINATE_DERIVATIVE_NORM	251
6.11.2.9 COORDINATE_INTERPOLATION_ADJUST	252
6.11.2.10 COORDINATE_INTERPOLATION_PARAMETERS_ADJUST	252
6.11.2.11 COORDINATE_METRICS_CALCULATE	253
6.11.2.12 COORDINATE_SYSTEM_CREATE_FINISH	253
6.11.2.13 COORDINATE_SYSTEM_CREATE_START	253
6.11.2.14 COORDINATE_SYSTEM_DESTROY_NUMBER	253
6.11.2.15 COORDINATE_SYSTEM_DESTROY_PTR	254
6.11.2.16 COORDINATE_SYSTEM_DIMENSION_GET	254
6.11.2.17 COORDINATE_SYSTEM_DIMENSION_SET_NUMBER	254
6.11.2.18 COORDINATE_SYSTEM_DIMENSION_SET_PTR	254
6.11.2.19 COORDINATE_SYSTEM_FOCUS_GET	255
6.11.2.20 COORDINATE_SYSTEM_FOCUS_SET_NUMBER	255
6.11.2.21 COORDINATE_SYSTEM_FOCUS_SET_PTR	255
6.11.2.22 COORDINATE_SYSTEM_NORMAL_CALCULATE	255
6.11.2.23 COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER	256
6.11.2.24 COORDINATE_SYSTEM_ORIENTATION_SET_PTR	256

6.11.2.25 COORDINATE_SYSTEM_ORIGIN_SET_NUMBER	256
6.11.2.26 COORDINATE_SYSTEM_ORIGIN_SET_PTR	256
6.11.2.27 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_GET	257
6.11.2.28 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER	257
6.11.2.29 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR	257
6.11.2.30 COORDINATE_SYSTEM_TYPE_GET	257
6.11.2.31 COORDINATE_SYSTEM_TYPE_SET_NUMBER	258
6.11.2.32 COORDINATE_SYSTEM_TYPE_SET_PTR	258
6.11.2.33 COORDINATE_SYSTEM_USER_NUMBER_FIND	258
6.11.2.34 COORDINATE_SYSTEMS_FINALISE	259
6.11.2.35 COORDINATE_SYSTEMS_INITIALISE	259
6.11.2.36 D2ZX_DP	259
6.11.2.37 DXZ_DP	259
6.11.2.38 DZX_DP	260
6.11.3 Variable Documentation	260
6.11.3.1 _SYSTEM_TYPE_STRING	260
6.11.3.2 COORDINAT	260
6.11.3.3 COORDINATE_SYSTEMS	260
6.11.3.4 GLOBAL_COORDINATE_SYSTEM	260
6.12 DOMAIN_MAPPINGS Namespace Reference	261
6.12.1 Detailed Description	261
6.12.2 Function Documentation	262
6.12.2.1 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE	262
6.12.2.2 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE	262
6.12.2.3 DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE	262
6.12.2.4 DOMAIN_MAPPINGS_MAPPING_FINALISE	263
6.12.2.5 DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE	263
6.12.2.6 DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE	264
6.12.2.7 DOMAIN_MAPPINGS_MAPPING_INITIALISE	264
6.13 ELASTICITY_ROUTINES Namespace Reference	265
6.13.1 Detailed Description	265
6.14 ELECTROMECHANICS_ROUTINES Namespace Reference	266
6.14.1 Detailed Description	266
6.15 EQUATIONS_SET_CONSTANTS Namespace Reference	267
6.15.1 Detailed Description	269

6.15.2	Variable Documentation	270
6.15.2.1	EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE	270
6.15.2.2	EQUATIONS_SET_BIHAMONIC_EQUATION_TYPE	270
6.15.2.3	EQUATIONS_SET_CLASSICAL_FIELD_CLASS	270
6.15.2.4	EQUATIONS_SET_DIFFUSION_EQUATION_TYPE	270
6.15.2.5	EQUATIONS_SET_ELASTICITY_CLASS	270
6.15.2.6	EQUATIONS_SET_ELECTROMAGNETICS_CLASS	270
6.15.2.7	EQUATIONS_SET_ELECTROSTATIC_TYPE	271
6.15.2.8	EQUATIONS_SETFINITE_ELASTICITY_TYPE	271
6.15.2.9	EQUATIONS_SET_FITTING_CLASS	271
6.15.2.10	EQUATIONS_SET_FLUID_MECHANICS_CLASS	271
6.15.2.11	EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE	271
6.15.2.12	EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE	271
6.15.2.13	EQUATIONS_SET_LAPLACE_EQUATION_TYPE	272
6.15.2.14	EQUATIONS_SET_LINEAR_ELASTIC_MODAL_TYPE	272
6.15.2.15	EQUATIONS_SET_LINEAR_ELASTICITY_TYPE	272
6.15.2.16	EQUATIONS_SET_MAGNETOSTATIC_TYPE	272
6.15.2.17	EQUATIONS_SET_MAXWELLS_EQUATIONS_TYPE	272
6.15.2.18	EQUATIONS_SET_MODAL_CLASS	272
6.15.2.19	EQUATIONS_SET_NAVIER_STOKES_FLUID_TYPE	272
6.15.2.20	EQUATIONS_SET_NO_CLASS	272
6.15.2.21	EQUATIONS_SET_NO_SUBTYPE	273
6.15.2.22	EQUATIONS_SET_NO_TYPE	273
6.15.2.23	EQUATIONS_SET_OPTIMISATION_CLASS	273
6.15.2.24	EQUATIONS_SET_POISSON_EQUATION_TYPE	273
6.15.2.25	EQUATIONS_SETREACTION_DIFFUSION_EQUATION_TYPE	273
6.15.2.26	EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE	273
6.15.2.27	EQUATIONS_SET_STOKES_FLUID_TYPE	274
6.15.2.28	EQUATIONS_SET_WAVE_EQUATION_TYPE	274
6.16	EQUATIONS_SET_ROUTINES Namespace Reference	275
6.16.1	Detailed Description	280
6.16.2	Function Documentation	280
6.16.2.1	EQ	280
6.16.2.2	EQUATIONS_INTERPOLATION_FINALISE	281
6.16.2.3	EQUATIONS_INTERPOLATION_INITIALISE	281
6.16.2.4	EQUATIONS_SET_ANALYTIC_CREATE_FINISH	281

6.16.2.5 EQUATIONS_SET_ANALYTIC_CREATE_START	282
6.16.2.6 EQUATIONS_SET_ANALYTIC_DESTROY	282
6.16.2.7 EQUATIONS_SET_ANALYTIC_FINALISE	283
6.16.2.8 EQUATIONS_SET_ANALYTIC_INITIALISE	283
6.16.2.9 EQUATIONS_SET_ASSEMBLE	283
6.16.2.10 EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM	284
6.16.2.11 EQUATIONS_SET_BACKSUBSTITUTE	284
6.16.2.12 EQUATIONS_SET_CREATE_FINISH	285
6.16.2.13 EQUATIONS_SET_CREATE_START	285
6.16.2.14 EQUATIONS_SET_DEPENDENT_	286
6.16.2.15 EQUATIONS_SET_DEPENDENT_CREATE_FINISH	286
6.16.2.16 EQUATIONS_SET_DEPENDENT_CREATE_START	287
6.16.2.17 EQUATIONS_SET_DEPENDENT_DEPENDENT_FIELD_GET	287
6.16.2.18 EQUATIONS_SET_DEPENDENT_DESTROY	288
6.16.2.19 EQUATIONS_SET_DEPENDENT_FINALISE	288
6.16.2.20 EQUATIONS_SET_DEPENDENT_INITIALISE	288
6.16.2.21 EQUATIONS_SET_DEPENDENT_SCALING_SET	289
6.16.2.22 EQUATIONS_SET_DESTROY	289
6.16.2.23 EQUATIONS_SET_EQUATIONS_CREATE_FINISH	289
6.16.2.24 EQUATIONS_SET_EQUATIONS_CREATE_START	290
6.16.2.25 EQUATIONS_SET_EQUATIONS_FINALISE	290
6.16.2.26 EQUATIONS_SET_EQUATIONS_INITIALISE	291
6.16.2.27 EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE	291
6.16.2.28 EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE	291
6.16.2.29 EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE	292
6.16.2.30 EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_INITIALISE	292
6.16.2.31 EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET	292
6.16.2.32 EQUATIONS_SET_EQUATIONS_SPARSITY_TYPE_SET	293
6.16.2.33 EQUATIONS_SET_EQUATIONS_TIME_DATA_FINALISE	293
6.16.2.34 EQUATIONS_SET_EQUATIONS_TIME_DATA_INITIALISE	294
6.16.2.35 EQUATIONS_SET_FINALISE	294
6.16.2.36 EQUATIONS_SETFINITE_ELEMENT_CALCULATE	295
6.16.2.37 EQUATIONS_SET_FIXED_CONDITIONS_APPLY	295
6.16.2.38 EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH	296
6.16.2.39 EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START	296
6.16.2.40 EQUATIONS_SET_FIXED_CONDITIONS_DESTROY	296

6.16.2.41 EQUATIONS_SET_FIXED_CONDITIONS_FINALISE	297
6.16.2.42 EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE	297
6.16.2.43 EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1	298
6.16.2.44 EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS	298
6.16.2.45 EQUATIONS_SET_GEOMETRY_FINALISE	299
6.16.2.46 EQUATIONS_SET_GEOMETRY_INITIALISE	299
6.16.2.47 EQUATIONS_SET_INITIALISE	299
6.16.2.48 EQUATIONS_SET_MATERIALS_COMPONENT_- INTERPOLATION_SET	300
6.16.2.49 EQUATIONS_SET_MATERIALS_COMPONENT_MESH_- COMPONENT_SET	300
6.16.2.50 EQUATIONS_SET_MATERIALS_CREATE_FINISH	301
6.16.2.51 EQUATIONS_SET_MATERIALS_CREATE_START	301
6.16.2.52 EQUATIONS_SET_MATERIALS_DESTROY	302
6.16.2.53 EQUATIONS_SET_MATERIALS_FINALISE	302
6.16.2.54 EQUATIONS_SET_MATERIALS_INITIALISE	302
6.16.2.55 EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET	303
6.16.2.56 EQUATIONS_SET_MATERIALS_SCALING_SET	303
6.16.2.57 EQUATIONS_SET_SETUP	304
6.16.2.58 EQUATIONS_SET_SOURCE_CREATE_FINISH	304
6.16.2.59 EQUATIONS_SET_SOURCE_CREATE_START	305
6.16.2.60 EQUATIONS_SET_SOURCE_DESTROY	305
6.16.2.61 EQUATIONS_SET_SOURCE_FINALISE	305
6.16.2.62 EQUATIONS_SET_SOURCE_INITIALISE	306
6.16.2.63 EQUATIONS_SET_SOURCE_SCALING_SET	306
6.16.2.64 EQUATIONS_SET_SPECIFICAT	307
6.16.2.65 EQUATIONS_SET_USER_NUMBER_FIND	307
6.16.2.66 EQUATIONS_SETS_FINALISE	308
6.16.2.67 EQUATIONS_SETS_INITIALISE	308
6.17 F90C Namespace Reference	309
6.17.1 Detailed Description	309
6.17.2 Function Documentation	309
6.17.2.1 C2FSTRING	309
6.17.2.2 CSTRINGLENGTH	310
6.17.2.3 F2CSTRING	310
6.17.2.4 FSTRINGLENGTH	310
6.18 FIELD_IO_ROUTINES Namespace Reference	311

6.18.1	Detailed Description	314
6.18.2	Function Documentation	315
6.18.2.1	FIE	315
6.18.2.2	FIELD_IO_BASIS_LHTP_FAMILY_LABEL	315
6.18.2.3	FIELD_IO_CREATE_FIELDS	315
6.18.2.4	FIELD_IO_DERIVATIVE_INFO	316
6.18.2.5	FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS	316
6.18.2.6	FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE	316
6.18.2.7	FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE	317
6.18.2.8	FIELD_IO_ELEMENTAL_INFO_SET_SORT	317
6.18.2.9	FIELD_IO_ELEMENTS_EXPORT	318
6.18.2.10	FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN	318
6.18.2.11	FIELD_IO_EXPORT_ELEMENTS_INTO_	319
6.18.2.12	FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA	319
6.18.2.13	FIELD_IO_EXPORT_NODES_INTO_LOC	320
6.18.2.14	FIELD_IO_FIELD_INFO	320
6.18.2.15	FIELD_IO_FILEDS_GROUP_INFO_GET	320
6.18.2.16	FIELD_IO_FILEDS_IMPORT	321
6.18.2.17	FIELD_IO_FILL_BASIS_INFO	321
6.18.2.18	FIELD_IO_FORTRAN_FILE_CLOSE	322
6.18.2.19	FIELD_IO_FORTRAN_FILE_OPEN	322
6.18.2.20	FIELD_IO_FORTRAN_FILE_READ_DP	323
6.18.2.21	FIELD_IO_FORTRAN_FILE_READ_INTG	323
6.18.2.22	FIELD_IO_FORTRAN_FILE_READ_STRING	323
6.18.2.23	FIELD_IO_FORTRAN_FILE_REWIND	324
6.18.2.24	FIELD_IO_FORTRAN_FILE_WRITE_DP	324
6.18.2.25	FIELD_IO_FORTRAN_FILE_WRITE_INTG	325
6.18.2.26	FIELD_IO_FORTRAN_FILE_WRITE_STRING	325
6.18.2.27	FIELD_IO_IMPORT_GLOBAL_MESH	326
6.18.2.28	FIELD_IO_LABEL_DERIVATIVE_INFO_GET	326
6.18.2.29	FIELD_IO_LABEL_FIELD_INFO_GET	327
6.18.2.30	FIELD_IO_MULTI_FILES_INFO_GET	327
6.18.2.31	FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS	327
6.18.2.32	FIELD_IO_NODAL_INFO_SET_FINALIZE	328
6.18.2.33	FIELD_IO_NODAL_INFO_SET_INITIALISE	328
6.18.2.34	FIELD_IO_NODAL_INFO_SET_SORT	329

6.18.2.35 FIELD_IO_NODES_EXPORT	329
6.18.2.36 FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE	329
6.18.2.37 STRING_TO_MUTI_INTEGERS_VS	330
6.18.2.38 STRING_TO_MUTI_REALS_VS	330
6.18.3 Variable Documentation	330
6.18.3.1 FIELD_IO_COMPONENT_LABEL	330
6.18.3.2 FIELD_IO_DERIVATIVE_LABEL	330
6.18.3.3 FIELD_IO_FIELD_LABEL	331
6.18.3.4 FIELD_IO_SCALE_FACTORS_NUMBER_TYPE	331
6.18.3.5 FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE	331
6.18.3.6 FIELD_IO_VARIABLE_LABEL	331
6.18.3.7 SHAPE_SIZE	331
6.19 FIELD_ROUTINES Namespace Reference	332
6.19.1 Detailed Description	336
6.19.2 Function Documentation	336
6.19.2.1 FI	336
6.19.2.2 FIELD_COMPONENT_INTER	337
6.19.2.3 FIELD_COMPONENT_MESH_COM	337
6.19.2.4 FIELD_CREATE_FINISH	338
6.19.2.5 FIELD_CREATE_VALUES_CACHE_FINALISE	338
6.19.2.6 FIELD_CREATE_VALUES_CACHE_INITIALISE	339
6.19.2.7 FIELD_DEPENDENT_TYPE_SET_NUMBER	339
6.19.2.8 FIELD_DEPENDENT_TYPE_SET_PTR	340
6.19.2.9 FIELD_DESTROY	340
6.19.2.10 FIELD_DIMENSION_SET_NUMBER	341
6.19.2.11 FIELD_DIMENSION_SET_PTR	341
6.19.2.12 FIELD_INTERPOLATE_GAUSS	342
6.19.2.13 FIELD_INTERPOLATE_XI	343
6.19.2.14 FIELD_INTERPOLATED_POINT_FINALISE	343
6.19.2.15 FIELD_INTERPOLATED_POINT_INITIALISE	344
6.19.2.16 FIELD_INTERPOLATED_POINT_METRICS_CALCULATE	344
6.19.2.17 FIELD_INTERPOLATED_POINT_METRICS_FINALISE	345
6.19.2.18 FIELD_INTERPOLATED_POINT_METRICS_INITIALISE	345
6.19.2.19 FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET	346
6.19.2.20 FIELD_INTERPOLATION_PARAMETERS_FINALISE	346
6.19.2.21 FIELD_INTERPOLATION_PARAMETERS_INITIALISE	347

6.19.2.22 FIELD_INTERPOLATION_PARAMETERS_LINE_GET	347
6.19.2.23 FIELD_VARIABLE_COMPONENT_FINALISE	348
6.19.2.24 FIELD_VARIABLE_COMPONENT_INITIALISE	349
6.19.2.25 FIELD_VARIABLES_FINALISE	349
6.19.2.26 FIELD_VARIABLES_INITIALISE	349
6.19.2.27 FIELDS_FINALISE	350
6.19.2.28 FIELDS_INITIALISE	350
6.20 FINITE_ELEMENT_ROUTINES Namespace Reference	352
6.20.1 Detailed Description	352
6.20.2 Function Documentation	352
6.20.2.1 FEM_ELEMENT_MATRICES_FINALISE	352
6.20.2.2 FEM_ELEMENT_MATRICES_INITIALISE	352
6.21 FLUID_MECHANICS_ROUTINES Namespace Reference	353
6.21.1 Detailed Description	353
6.22 GENERATED_MESH_ROUTINES Namespace Reference	354
6.22.1 Detailed Description	354
6.22.2 Function Documentation	355
6.22.2.1 GENERATED_MESH_CREATE_FINISH	355
6.22.2.2 GENERATED_MESH_CREATE_START	355
6.22.2.3 GENERATED_MESH_DESTROY	355
6.22.2.4 GENERATED_MESH_FINALISE	356
6.22.2.5 GENERATED_MESH_INITIALISE	356
6.22.2.6 GENERATED_MESH_TYPE_SET	357
6.23 INPUT_OUTPUT Namespace Reference	358
6.23.1 Detailed Description	371
6.23.2 Function Documentation	372
6.23.2.1 WR	372
6.23.2.2 WR	372
6.23.2.3 WR	373
6.23.2.4 WR	374
6.23.2.5 WR	375
6.23.2.6 WR	375
6.23.2.7 WR	376
6.23.2.8 WR	377
6.23.2.9 WR	377
6.23.2.10 WR	378

6.23.2.11 WR	378
6.23.2.12 WR	379
6.23.2.13 WR	380
6.23.2.14 WR	380
6.23.2.15 WR	381
6.23.2.16 WR	381
6.23.2.17 WR	382
6.23.2.18 WR	383
6.23.2.19 WR	383
6.23.2.20 WR	384
6.23.2.21 WR	384
6.23.2.22 WR	385
6.23.2.23 WR	386
6.23.2.24 WR	386
6.23.2.25 WR	387
6.23.2.26 WR	387
6.23.2.27 WR	388
6.23.2.28 WR	389
6.23.2.29 WR	389
6.23.2.30 WR	390
6.23.2.31 WR	390
6.23.2.32 WR	391
6.23.2.33 WR	392
6.23.2.34 WR	392
6.23.2.35 WRITE_STRING_C	393
6.23.2.36 WRITE_STRING_FMT	393
6.23.2.37 WRITE_STRING_FMT	394
6.23.2.38 WRITE_STRING_FMT	395
6.23.2.39 WRITE_STRING_FMT	395
6.23.2.40 WRITE_STRING_FMT	396
6.23.2.41 WRITE_STRING_FMT	397
6.23.2.42 WRITE_STRING_FMT	397
6.23.2.43 WRITE_STRING_FMT_VALUE_C	398
6.23.2.44 WRITE_STRING_FMT_VALUE_DP	398
6.23.2.45 WRITE_STRING_FMT_VALUE_INTG	399
6.23.2.46 WRITE_STRING_FMT_VALUE_L	399

6.23.2.47 WRITE_STRING_FMT_VALUE_LINTG	400
6.23.2.48 WRITE_STRING_FMT_VALUE_SP	400
6.23.2.49 WRITE_STRING_FMT_VALUE_VS	401
6.23.2.50 WRITE_STRING_IDX	402
6.23.2.51 WRITE_STRING_IDX	402
6.23.2.52 WRITE_STRING_IDX	403
6.23.2.53 WRITE_STRING_IDX	404
6.23.2.54 WRITE_STRING_IDX	405
6.23.2.55 WRITE_STRING_MATRIX_DP	406
6.23.2.56 WRITE_STRING_MATRIX_INTG	407
6.23.2.57 WRITE_STRING_MATRIX_L	408
6.23.2.58 WRITE_STRING_MATRIX_LINTG	409
6.23.2.59 WRITE_STRING_MATRIX_SP	410
6.23.2.60 WRITE_STRING_TWO_VALUE_C_C	412
6.23.2.61 WRITE_STRING_TWO_VALUE_C_DP	412
6.23.2.62 WRITE_STRING_TWO_VALUE_C_INTG	413
6.23.2.63 WRITE_STRING_TWO_VALUE_C_L	413
6.23.2.64 WRITE_STRING_TWO_VALUE_C_SP	414
6.23.2.65 WRITE_STRING_TWO_VALUE_C_VS	415
6.23.2.66 WRITE_STRING_TWO_VALUE_DP_C	415
6.23.2.67 WRITE_STRING_TWO_VALUE_DP_DP	416
6.23.2.68 WRITE_STRING_TWO_VALUE_DP_INTG	416
6.23.2.69 WRITE_STRING_TWO_VALUE_DP_L	417
6.23.2.70 WRITE_STRING_TWO_VALUE_DP_SP	417
6.23.2.71 WRITE_STRING_TWO_VALUE_DP_VS	418
6.23.2.72 WRITE_STRING_TWO_VALUE_INTG_C	419
6.23.2.73 WRITE_STRING_TWO_VALUE_INTG_DP	419
6.23.2.74 WRITE_STRING_TWO_VALUE_INTG_INTG	420
6.23.2.75 WRITE_STRING_TWO_VALUE_INTG_L	420
6.23.2.76 WRITE_STRING_TWO_VALUE_INTG_SP	421
6.23.2.77 WRITE_STRING_TWO_VALUE_INTG_VS	422
6.23.2.78 WRITE_STRING_TWO_VALUE_L_C	422
6.23.2.79 WRITE_STRING_TWO_VALUE_L_DP	423
6.23.2.80 WRITE_STRING_TWO_VALUE_L_INTG	423
6.23.2.81 WRITE_STRING_TWO_VALUE_L_L	424
6.23.2.82 WRITE_STRING_TWO_VALUE_L_SP	424

6.23.2.83 WRITE_STRING_TWO_VALUE_L_VS	425
6.23.2.84 WRITE_STRING_TWO_VALUE_SP_C	426
6.23.2.85 WRITE_STRING_TWO_VALUE_SP_DP	426
6.23.2.86 WRITE_STRING_TWO_VALUE_SP_INTG	427
6.23.2.87 WRITE_STRING_TWO_VALUE_SP_L	427
6.23.2.88 WRITE_STRING_TWO_VALUE_SP_SP	428
6.23.2.89 WRITE_STRING_TWO_VALUE_SP_VS	429
6.23.2.90 WRITE_STRING_TWO_VALUE_VS_C	429
6.23.2.91 WRITE_STRING_TWO_VALUE_VS_DP	430
6.23.2.92 WRITE_STRING_TWO_VALUE_VS_INTG	430
6.23.2.93 WRITE_STRING_TWO_VALUE_VS_L	431
6.23.2.94 WRITE_STRING_TWO_VALUE_VS_SP	432
6.23.2.95 WRITE_STRING_TWO_VALUE_VS_VS	432
6.23.2.96 WRITE_STRING_VALUE_C	433
6.23.2.97 WRITE_STRING_VALUE_DP	433
6.23.2.98 WRITE_STRING_VALUE_INTG	434
6.23.2.99 WRITE_STRING_VALUE_L	434
6.23.2.100 WRITE_STRING_VALUE_LINTG	435
6.23.2.101 WRITE_STRING_VALUE_SP	435
6.23.2.102 WRITE_STRING_VALUE_VS	436
6.23.2.103 WRITE_STRING_VS	436
6.24 ISO_VARYING_STRING Namespace Reference	437
6.24.1 Detailed Description	439
6.24.2 Function Documentation	439
6.24.2.1 adjustl_	439
6.24.2.2 adjustr_	440
6.24.2.3 char_auto	440
6.24.2.4 char_fixed	440
6.24.2.5 extract_CH	440
6.24.2.6 extract_VS	440
6.24.2.7 get_	440
6.24.2.8 get_set_CH	440
6.24.2.9 get_set_VS	440
6.24.2.10 get_unit	441
6.24.2.11 get_unit_set_CH	441
6.24.2.12 get_unit_set_VS	441

6.24.2.13 iachar_	441
6.24.2.14 ichar_	441
6.24.2.15 index_CH_VS	441
6.24.2.16 index_VS_CH	441
6.24.2.17 index_VS_VS	441
6.24.2.18 insert_CH_CH	442
6.24.2.19 insert_CH_VS	442
6.24.2.20 insert_VS_CH	442
6.24.2.21 insert_VS_VS	442
6.24.2.22 len_	442
6.24.2.23 len_trim_	442
6.24.2.24 lge_CH_VS	442
6.24.2.25 lge_VS_CH	442
6.24.2.26 lge_VS_VS	442
6.24.2.27 lgt_CH_VS	443
6.24.2.28 lgt_VS_CH	443
6.24.2.29 lgt_VS_VS	443
6.24.2.30 lle_CH_VS	443
6.24.2.31 lle_VS_CH	443
6.24.2.32 lle_VS_VS	443
6.24.2.33 llt_CH_VS	443
6.24.2.34 llt_VS_CH	443
6.24.2.35 llt_VS_VS	443
6.24.2.36 op_assign_CH_VS	444
6.24.2.37 op_assign_VS_CH	444
6.24.2.38 op_concat_CH_VS	444
6.24.2.39 op_concat_VS_CH	444
6.24.2.40 op_concat_VS_VS	444
6.24.2.41 op_eq_CH_VS	444
6.24.2.42 op_eq_VS_CH	444
6.24.2.43 op_eq_VS_VS	445
6.24.2.44 op_ge_CH_VS	445
6.24.2.45 op_ge_VS_CH	445
6.24.2.46 op_ge_VS_VS	445
6.24.2.47 op_gt_CH_VS	445
6.24.2.48 op_gt_VS_CH	445

6.24.2.49 op_gt_VS_VS	445
6.24.2.50 op_le_CH_VS	445
6.24.2.51 op_le_VS_CH	445
6.24.2.52 op_le_VS_VS	446
6.24.2.53 op_lt_CH_VS	446
6.24.2.54 op_lt_VS_CH	446
6.24.2.55 op_lt_VS_VS	446
6.24.2.56 op_ne_CH_VS	446
6.24.2.57 op_ne_VS_CH	446
6.24.2.58 op_ne_VS_VS	446
6.24.2.59 put_CH	446
6.24.2.60 put_line_CH	446
6.24.2.61 put_line_unit_CH	447
6.24.2.62 put_line_unit_VS	447
6.24.2.63 put_line_VS	447
6.24.2.64 put_unit_CH	447
6.24.2.65 put_unit_VS	447
6.24.2.66 put_VS	447
6.24.2.67 remove_CH	447
6.24.2.68 remove_VS	447
6.24.2.69 repeat_	447
6.24.2.70 replace_CH_CH_auto	448
6.24.2.71 replace_CH_CH_CH_target	448
6.24.2.72 replace_CH_CH_fixed	448
6.24.2.73 replace_CH_CH_VS_target	448
6.24.2.74 replace_CH_VS_auto	448
6.24.2.75 replace_CH_VS_CH_target	448
6.24.2.76 replace_CH_VS_fixed	448
6.24.2.77 replace_CH_VS_VS_target	449
6.24.2.78 replace_VS_CH_auto	449
6.24.2.79 replace_VS_CH_CH_target	449
6.24.2.80 replace_VS_CH_fixed	449
6.24.2.81 replace_VS_CH_VS_target	449
6.24.2.82 replace_VS_VS_auto	449
6.24.2.83 replace_VS_VS_CH_target	449
6.24.2.84 replace_VS_VS_fixed	450

6.24.2.85 replace_VS_VS_VS_target	450
6.24.2.86 scan_CH_VS	450
6.24.2.87 scan_VS_CH	450
6.24.2.88 scan_VS_VS	450
6.24.2.89 split_CH	450
6.24.2.90 split_VS	450
6.24.2.91 trim_	451
6.24.2.92 var_str_	451
6.24.2.93 verify_CH_VS	451
6.24.2.94 verify_VS_CH	451
6.24.2.95 verify_VS_VS	451
6.24.3 Variable Documentation	451
6.24.3.1 GET_BUFFER_LEN	451
6.25 KINDS Namespace Reference	452
6.25.1 Detailed Description	452
6.26 LAPACK Namespace Reference	453
6.26.1 Detailed Description	453
6.27 LAPLACE_EQUATIONS_ROUTINES Namespace Reference	454
6.27.1 Detailed Description	454
6.27.2 Function Documentation	454
6.27.2.1 LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALECULATE	454
6.27.2.2 LAPLACE_EQUATION_EQUATIONS_SET_SETUP	455
6.27.2.3 LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP	456
6.27.2.4 LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET	457
6.27.2.5 LAPLACE_EQUATION_PROBLEM_SETUP	457
6.27.2.6 LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP	458
6.27.2.7 LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET	458
6.28 LISTS Namespace Reference	460
6.28.1 Detailed Description	464
6.28.2 Function Documentation	464
6.28.2.1 LIST_CREATE_FINISH	464
6.28.2.2 LIST_CREATE_START	464
6.28.2.3 LIST_DATA_TYPE_SET	465
6.28.2.4 LIST_DESTROY	465
6.28.2.5 LIST_DETACH_AND_DESTROY_DP	466
6.28.2.6 LIST_DETACH_AND_DESTROY_INTG	466

6.28.2.7 LIST_DETACH_AND_DESTROY_SP	467
6.28.2.8 LIST_FINALISE	467
6.28.2.9 LIST_INITIAL_SIZE_SET	468
6.28.2.10 LIST_INITIALISE	468
6.28.2.11 LIST_ITEM_ADD_DP1	469
6.28.2.12 LIST_ITEM_ADD_INTG1	469
6.28.2.13 LIST_ITEM_ADD_SP1	469
6.28.2.14 LIST_ITEM_DELETE	470
6.28.2.15 LIST_ITEM_IN_LIST_DP1	470
6.28.2.16 LIST_ITEM_IN_LIST_INTG1	471
6.28.2.17 LIST_ITEM_IN_LIST_SP1	471
6.28.2.18 LIST_NUMBER_OF_ITEMS_GET	471
6.28.2.19 LIST_REMOVE_DUPLICATES	472
6.28.2.20 LIST_SEARCH_DP_ARRAY	472
6.28.2.21 LIST_SEARCH_INTG_ARRAY	473
6.28.2.22 LIST_SEARCH_LINEAR_DP_ARRAY	473
6.28.2.23 LIST_SEARCH_LINEAR_INTG_ARRAY	474
6.28.2.24 LIST_SEARCH_LINEAR_SP_ARRAY	474
6.28.2.25 LIST_SEARCH_SP_ARRAY	475
6.28.2.26 LIST_SORT_BUBBLE_DP_ARRAY	475
6.28.2.27 LIST_SORT_BUBBLE_INTG_ARRAY	475
6.28.2.28 LIST_SORT_BUBBLE_SP_ARRAY	476
6.28.2.29 LIST_SORT_DP_ARRAY	476
6.28.2.30 LIST_SORT_HEAP_DP_ARRAY	477
6.28.2.31 LIST_SORT_HEAP_INTG_ARRAY	477
6.28.2.32 LIST_SORT_HEAP_SP_ARRAY	477
6.28.2.33 LIST_SORT_INTG_ARRAY	478
6.28.2.34 LIST_SORT_SHELL_DP_ARRAY	478
6.28.2.35 LIST_SORT_SHELL_INTG_ARRAY	478
6.28.2.36 LIST_SORT_SHELL_SP_ARRAY	479
6.28.2.37 LIST_SORT_SP_ARRAY	479
6.29 MACHINE_CONSTANTS Namespace Reference	480
6.29.1 Detailed Description	480
6.29.2 Variable Documentation	480
6.29.2.1 CHARACTER_SIZE	480
6.29.2.2 DOUBLE_COMPLEX_SIZE	480

6.29.2.3 DOUBLE_REAL_SIZE	481
6.29.2.4 ERROR_SEPARATOR_CONSTANT	481
6.29.2.5 INTEGER_SIZE	481
6.29.2.6 LOGICAL_SIZE	481
6.29.2.7 LONG_INTEGER_SIZE	481
6.29.2.8 MACHINE_CHAR_FORMAT	481
6.29.2.9 MACHINE_DP_FORMAT	481
6.29.2.10 MACHINE_ENDIAN	481
6.29.2.11 MACHINE_INT_FORMAT	482
6.29.2.12 MACHINE_OS	482
6.29.2.13 MACHINE_SP_FORMAT	482
6.29.2.14 MACHINE_TYPE	482
6.29.2.15 SHORT_INTEGER_SIZE	482
6.29.2.16 SINGLE_COMPLEX_SIZE	482
6.29.2.17 SINGLE_REAL_SIZE	483
6.30 MATHS Namespace Reference	484
6.30.1 Detailed Description	485
6.30.2 Function Documentation	485
6.30.2.1 CROSS_PRODUCT_DP	485
6.30.2.2 CROSS_PRODUCT_INTG	485
6.30.2.3 CROSS_PRODUCT_SP	485
6.30.2.4 D_CROSS_PRODUCT_DP	486
6.30.2.5 D_CROSS_PRODUCT_INTG	486
6.30.2.6 D_CROSS_PRODUCT_SP	486
6.30.2.7 DETERMINANT_FULL_DP	486
6.30.2.8 DETERMINANT_FULL_INTG	487
6.30.2.9 DETERMINANT_FULL_SP	487
6.30.2.10 EDP_DP	487
6.30.2.11 EDP_SP	487
6.30.2.12 EIGENVALUE_FULL_DP	487
6.30.2.13 EIGENVALUE_FULL_SP	487
6.30.2.14 EIGENVECTOR_FULL_DP	488
6.30.2.15 EIGENVECTOR_FULL_SP	488
6.30.2.16 I0_DP	488
6.30.2.17 I0_SP	488
6.30.2.18 I1_DP	488

6.30.2.19 I1_SP	488
6.30.2.20 INVERT_FULL_DP	488
6.30.2.21 INVERT_FULL_SP	489
6.30.2.22 K0_DP	489
6.30.2.23 K0_SP	489
6.30.2.24 K1_DP	489
6.30.2.25 K1_SP	489
6.30.2.26 KDP_DP	489
6.30.2.27 KDP_SP	489
6.30.2.28 L2NORM_DP	490
6.30.2.29 L2NORM_SP	490
6.30.2.30 NORMALISE_DP	490
6.30.2.31 NORMALISE_SP	490
6.30.2.32 SOLVE_SMALL_LINEAR_SYSTEM_DP	490
6.30.2.33 SOLVE_SMALL_LINEAR_SYSTEM_SP	490
6.31 MATRIX_VECTOR Namespace Reference	491
6.31.1 Detailed Description	499
6.31.2 Function Documentation	499
6.31.2.1 MATRIX_ALL_VALUES_SET_DP	499
6.31.2.2 MATRIX_ALL_VALUES_SET_INTG	499
6.31.2.3 MATRIX_ALL_VALUES_SET_L	500
6.31.2.4 MATRIX_ALL_VALUES_SET_SP	500
6.31.2.5 MATRIX_CREATE_FINISH	501
6.31.2.6 MATRIX_CREATE_START	501
6.31.2.7 MATRIX_DATA_GET_DP	501
6.31.2.8 MATRIX_DATA_GET_INTG	502
6.31.2.9 MATRIX_DATA_GET_L	502
6.31.2.10 MATRIX_DATA_GET_SP	503
6.31.2.11 MATRIX_DATA_TYPE_SET	503
6.31.2.12 MATRIX_DESTROY	504
6.31.2.13 MATRIX_DUPLICATE	504
6.31.2.14 MATRIX_FINALISE	504
6.31.2.15 MATRIX_INITIALISE	505
6.31.2.16 MATRIX_MAX_COLUMNS_PER_ROW_GET	505
6.31.2.17 MATRIX_MAX_SIZE_SET	506
6.31.2.18 MATRIX_NUMBER_NON_ZEROS_SET	506

6.31.2.19 MATRIX_OUTPUT	507
6.31.2.20 MATRIX_SIZE_SET	507
6.31.2.21 MATRIX_STORAGE_LOCATION_FIND	508
6.31.2.22 MATRIX_STORAGE_LOCATIONS_GET	508
6.31.2.23 MATRIX_STORAGE_LOCATIONS_SET	509
6.31.2.24 MATRIX_STORAGE_TYPE_GET	509
6.31.2.25 MATRIX_STORAGE_TYPE_SET	510
6.31.2.26 MATRIX_VALUES_ADD_DP	510
6.31.2.27 MATRIX_VALUES_ADD_DP1	511
6.31.2.28 MATRIX_VALUES_ADD_DP2	511
6.31.2.29 MATRIX_VALUES_ADD_INTG	512
6.31.2.30 MATRIX_VALUES_ADD_INTG1	512
6.31.2.31 MATRIX_VALUES_ADD_INTG2	513
6.31.2.32 MATRIX_VALUES_ADD_L	513
6.31.2.33 MATRIX_VALUES_ADD_L1	514
6.31.2.34 MATRIX_VALUES_ADD_L2	514
6.31.2.35 MATRIX_VALUES_ADD_SP	515
6.31.2.36 MATRIX_VALUES_ADD_SP1	515
6.31.2.37 MATRIX_VALUES_ADD_SP2	516
6.31.2.38 MATRIX_VALUES_GET_DP	516
6.31.2.39 MATRIX_VALUES_GET_DP1	517
6.31.2.40 MATRIX_VALUES_GET_DP2	517
6.31.2.41 MATRIX_VALUES_GET_INTG	518
6.31.2.42 MATRIX_VALUES_GET_INTG1	518
6.31.2.43 MATRIX_VALUES_GET_INTG2	519
6.31.2.44 MATRIX_VALUES_GET_L	519
6.31.2.45 MATRIX_VALUES_GET_L1	520
6.31.2.46 MATRIX_VALUES_GET_L2	520
6.31.2.47 MATRIX_VALUES_GET_SP	521
6.31.2.48 MATRIX_VALUES_GET_SP1	521
6.31.2.49 MATRIX_VALUES_GET_SP2	522
6.31.2.50 MATRIX_VALUES_SET_DP	522
6.31.2.51 MATRIX_VALUES_SET_DP1	523
6.31.2.52 MATRIX_VALUES_SET_DP2	523
6.31.2.53 MATRIX_VALUES_SET_INTG	524
6.31.2.54 MATRIX_VALUES_SET_INTG1	524

6.31.2.55 MATRIX_VALUES_SET_INTG2	525
6.31.2.56 MATRIX_VALUES_SET_L	525
6.31.2.57 MATRIX_VALUES_SET_L1	526
6.31.2.58 MATRIX_VALUES_SET_L2	526
6.31.2.59 MATRIX_VALUES_SET_SP	527
6.31.2.60 MATRIX_VALUES_SET_SP1	527
6.31.2.61 MATRIX_VALUES_SET_SP2	528
6.31.2.62 VECTOR_ALL_VALUES_SET_DP	528
6.31.2.63 VECTOR_ALL_VALUES_SET_INTG	529
6.31.2.64 VECTOR_ALL_VALUES_SET_L	529
6.31.2.65 VECTOR_ALL_VALUES_SET_SP	529
6.31.2.66 VECTOR_CREATE_FINISH	530
6.31.2.67 VECTOR_CREATE_START	530
6.31.2.68 VECTOR_DATA_GET_DP	531
6.31.2.69 VECTOR_DATA_GET_INTG	531
6.31.2.70 VECTOR_DATA_GET_L	531
6.31.2.71 VECTOR_DATA_GET_SP	532
6.31.2.72 VECTOR_DATA_TYPE_SET	532
6.31.2.73 VECTOR_DESTROY	533
6.31.2.74 VECTOR_DUPLICATE	533
6.31.2.75 VECTOR_FINALISE	534
6.31.2.76 VECTOR_INITIALISE	534
6.31.2.77 VECTOR_SIZE_SET	534
6.31.2.78 VECTOR_VALUES_GET_DP	535
6.31.2.79 VECTOR_VALUES_GET_DP1	535
6.31.2.80 VECTOR_VALUES_GET_INTG	536
6.31.2.81 VECTOR_VALUES_GET_INTG1	536
6.31.2.82 VECTOR_VALUES_GET_L	536
6.31.2.83 VECTOR_VALUES_GET_L1	537
6.31.2.84 VECTOR_VALUES_GET_SP	537
6.31.2.85 VECTOR_VALUES_GET_SP1	538
6.31.2.86 VECTOR_VALUES_SET_DP	538
6.31.2.87 VECTOR_VALUES_SET_DP1	538
6.31.2.88 VECTOR_VALUES_SET_INTG	539
6.31.2.89 VECTOR_VALUES_SET_INTG1	539
6.31.2.90 VECTOR_VALUES_SET_L	540

6.31.2.91 VECTOR_VALUES_SET_L1	540
6.31.2.92 VECTOR_VALUES_SET_SP	540
6.31.2.93 VECTOR_VALUES_SET_SP1	541
6.31.3 Variable Documentation	541
6.31.3.1 MATRIX_VECTOR_ID	541
6.32 MESH_ROUTINES Namespace Reference	542
6.32.1 Detailed Description	547
6.32.2 Function Documentation	547
6.32.2.1 DECOMPOSITION_CREATE_FINISH	547
6.32.2.2 DECOMPOSITION_CREATE_START	547
6.32.2.3 DECOMPOSITION_DESTROY	548
6.32.2.4 DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE	548
6.32.2.5 DECOMPOSITION_ELEMENT_DOMAIN_SET	549
6.32.2.6 DECOMPOSITION_MESH_COMPONENT_NUMBER_SET	549
6.32.2.7 DECOMPOSITION_NUMBER_OF_DOMAINS_SET	550
6.32.2.8 DOMAIN_MAPPINGS_NODES_FINALISE	550
6.32.2.9 DOMAIN_MAPPINGS_NODES_INITIALISE	551
6.32.2.10 DOMAIN_TOPOLOGY_CALCULATE	551
6.32.2.11 DOMAIN_TOPOLOGY_DOFS_FINALISE	552
6.32.2.12 DOMAIN_TOPOLOGY_DOFS_INITIALISE	552
6.32.2.13 DOMAIN_TOPOLOGY_ELEMENT_FINALISE	553
6.32.2.14 DOMAIN_TOPOLOGY_ELEMENT_INITIALISE	553
6.32.2.15 DOMAIN_TOPOLOGY_ELEMENTS_FINALISE	554
6.32.2.16 DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE	554
6.32.2.17 DOMAIN_TOPOLOGY_FINALISE	555
6.32.2.18 DOMAIN_TOPOLOGY_INITIALISE	555
6.32.2.19 DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH	556
6.32.2.20 DOMAIN_TOPOLOGY_LINE_FINALISE	556
6.32.2.21 DOMAIN_TOPOLOGY_LINE_INITIALISE	556
6.32.2.22 DOMAIN_TOPOLOGY_LINES_FINALISE	557
6.32.2.23 DOMAIN_TOPOLOGY_LINES_INITIALISE	557
6.32.2.24 DOMAIN_TOPOLOGY_NODE_FINALISE	558
6.32.2.25 DOMAIN_TOPOLOGY_NODE_INITIALISE	558
6.32.2.26 DOMAIN_TOPOLOGY_NODES_FINALISE	559
6.32.2.27 DOMAIN_TOPOLOGY_NODES_INITIALISE	559
6.32.2.28 DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_- CALCULATE	560

6.32.2.29 MESH_CREATE_FINISH	560
6.32.2.30 MESH_CREATE_REGULAR	561
6.32.2.31 MESH_CREATE_START	561
6.32.2.32 MESH_DESTROY	562
6.32.2.33 MESH_FINALISE	562
6.32.2.34 MESH_INITIALISE	563
6.32.2.35 MESH_NUMBER_OF_COMPONENTS_SET_NUMBER	563
6.32.2.36 MESH_NUMBER_OF_COMPONENTS_SET_PTR	564
6.32.2.37 MESH_NUMBER_OF_ELEMENTS_SET_NUMBER	564
6.32.2.38 MESH_NUMBER_OF_ELEMENTS_SET_PTR	565
6.32.2.39 MESH_TOPOLOGY_CALCULATE	565
6.32.2.40 MESH_TOPOLOGY_DOFS_CALCULATE	565
6.32.2.41 MESH_TOPOLOGY_DOFS_FINALISE	566
6.32.2.42 MESH_TOPOLOGY_DOFS_INITIALISE	566
6.32.2.43 MESH_TOPOLOGY_ELEMENT_FINALISE	567
6.32.2.44 MESH_TOPOLOGY_ELEMENT_INITIALISE	567
6.32.2.45 MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_- CALCULATE	568
6.32.2.46 MESH_TOPOLOGY_ELEMENTS_BASIS_SET	568
6.32.2.47 MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH	569
6.32.2.48 MESH_TOPOLOGY_ELEMENTS_CREATE_START	569
6.32.2.49 MESH_TOPOLOGY_ELEMENTS_DESTROY	570
6.32.2.50 MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET	570
6.32.2.51 MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET	571
6.32.2.52 MESH_TOPOLOGY_ELEMENTS_FINALISE	571
6.32.2.53 MESH_TOPOLOGY_ELEMENTS_INITIALISE	572
6.32.2.54 MESH_TOPOLOGY_ELEMENTS_NUMBER_SET	572
6.32.2.55 MESH_TOPOLOGY_FINALISE	573
6.32.2.56 MESH_TOPOLOGY_INITIALISE	573
6.32.2.57 MESH_TOPOLOGY_NODE_FINALISE	574
6.32.2.58 MESH_TOPOLOGY_NODE_INITIALISE	574
6.32.2.59 MESH_TOPOLOGY_NODES_CALCULATE	575
6.32.2.60 MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE	575
6.32.2.61 MESH_TOPOLOGY_NODES_FINALISE	576
6.32.2.62 MESH_TOPOLOGY_NODES_INITIALISE	576
6.32.2.63 MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_- CALCULATE	577

6.32.2.64 MESH_USER_NUMBER_FIND	577
6.32.2.65 MESHES_FINALISE	578
6.32.2.66 MESHES_INITIALISE	578
6.33 NODE_ROUTINES Namespace Reference	579
6.33.1 Detailed Description	579
6.33.2 Function Documentation	579
6.33.2.1 NODE_CHECK_EXISTS	579
6.33.2.2 NODE_DESTROY	579
6.33.2.3 NODE_INITIAL_POSITION_SET	580
6.33.2.4 NODE_NUMBER_SET	580
6.33.2.5 NODES_CREATE_FINISH	580
6.33.2.6 NODES_CREATE_START	580
6.33.2.7 NODES_FINALISE	581
6.33.2.8 NODES_INITIALISE	581
6.34 PROBLEM_CONSTANTS Namespace Reference	582
6.34.1 Detailed Description	583
6.34.2 Variable Documentation	583
6.34.2.1 PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE	583
6.34.2.2 PROBLEM_BIHAMONIC_EQUATION_TYPE	583
6.34.2.3 PROBLEM_CLASSICAL_FIELD_CLASS	584
6.34.2.4 PROBLEM_DIFFUSION_EQUATION_TYPE	584
6.34.2.5 PROBLEM_ELASTICITY_CLASS	584
6.34.2.6 PROBLEM_ELECTROMAGNETICS_CLASS	584
6.34.2.7 PROBLEM_ELECTROSTATIC_TYPE	584
6.34.2.8 PROBLEMFINITE_ELASTICITY_TYPE	584
6.34.2.9 PROBLEM_FITTING_CLASS	584
6.34.2.10 PROBLEM_FLUID_MECHANICS_CLASS	585
6.34.2.11 PROBLEM_GENERALISED_LAPLACE_SUBTYPE	585
6.34.2.12 PROBLEM_HELMHOLTZ_EQUATION_TYPE	585
6.34.2.13 PROBLEM_LAPLACE_EQUATION_TYPE	585
6.34.2.14 PROBLEM_LINEAR_ELASTIC_MODAL_TYPE	585
6.34.2.15 PROBLEM_LINEAR_ELASTICITY_TYPE	585
6.34.2.16 PROBLEM_MAGNETOSTATIC_TYPE	585
6.34.2.17 PROBLEM_MAXWELLS_EQUATIONS_TYPE	586
6.34.2.18 PROBLEM_MODAL_CLASS	586
6.34.2.19 PROBLEM_NAVIER_STOKES_FLUID_TYPE	586

6.34.2.20 PROBLEM_NO_CLASS	586
6.34.2.21 PROBLEM_NO_SUBTYPE	586
6.34.2.22 PROBLEM_NO_TYPE	586
6.34.2.23 PROBLEM_OPTIMISATION_CLASS	586
6.34.2.24 PROBLEM_POISSON_EQUATION_TYPE	586
6.34.2.25 PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE	587
6.34.2.26 PROBLEM_STANDARD_LAPLACE_SUBTYPE	587
6.34.2.27 PROBLEM_STOKES_FLUID_TYPE	587
6.34.2.28 PROBLEM_WAVE_EQUATION_TYPE	587
6.35 PROBLEM_ROUTINES Namespace Reference	588
6.35.1 Detailed Description	590
6.35.2 Function Documentation	590
6.35.2.1 PROBLEM_CONTROL_CREATE_FINISH	590
6.35.2.2 PROBLEM_CONTROL_CREATE_START	591
6.35.2.3 PROBLEM_CONTROL_DESTROY	591
6.35.2.4 PROBLEM_CONTROL_FINALISE	592
6.35.2.5 PROBLEM_CONTROL_INITIALISE	592
6.35.2.6 PROBLEM_CREATE_FINISH	592
6.35.2.7 PROBLEM_CREATE_START	593
6.35.2.8 PROBLEM_DESTROY_NUMBER	593
6.35.2.9 PROBLEM_DESTROY_PTR	594
6.35.2.10 PROBLEM_FINALISE	594
6.35.2.11 PROBLEM_INITIALISE	594
6.35.2.12 PROBLEM_SETUP	595
6.35.2.13 PROBLEM SOLUTION EQUATIONS_SET_ADD	596
6.35.2.14 PROBLEM SOLUTION_FINALISE	596
6.35.2.15 PROBLEM SOLUTION_INITIALISE	596
6.35.2.16 PROBLEM SOLUTION_SOLVE	597
6.35.2.17 PROBLEM SOLUTIONS_CREATE_FINISH	597
6.35.2.18 PROBLEM SOLUTIONS_CREATE_START	598
6.35.2.19 PROBLEM SOLUTIONS_FINALISE	598
6.35.2.20 PROBLEM SOLUTIONS_INITIALISE	599
6.35.2.21 PROBLEM_SOLVE	599
6.35.2.22 PROBLEM_SOLVER_CREATE_FINISH	599
6.35.2.23 PROBLEM_SOLVER_CREATE_START	600
6.35.2.24 PROBLEM_SOLVER_DESTROY	600

6.35.2.25 PROBLEM_SOLVER_GET	600
6.35.2.26 PROBLEM_SPECIFICATION_SET_NUMBER	601
6.35.2.27 PROBLEM_SPECIFICATION_SET_PTR	601
6.35.2.28 PROBLEM_USER_NUMBER_FIND	602
6.35.2.29 PROBLEMS_FINALISE	602
6.35.2.30 PROBLEMS_INITIALISE	603
6.35.3 Variable Documentation	603
6.35.3.1 PROBLEMS	603
6.36 REGION_ROUTINES Namespace Reference	604
6.36.1 Detailed Description	604
6.36.2 Function Documentation	604
6.36.2.1 REGION_COORDINATE_SYSTEM_GET	604
6.36.2.2 REGION_COORDINATE_SYSTEM_SET_NUMBER	605
6.36.2.3 REGION_COORDINATE_SYSTEM_SET_PTR	605
6.36.2.4 REGION_CREATE_FINISH	605
6.36.2.5 REGION_CREATE_START	605
6.36.2.6 REGION_DESTROY	606
6.36.2.7 REGION_LABEL_GET	606
6.36.2.8 REGION_LABEL_SET_NUMBER	606
6.36.2.9 REGION_LABEL_SET_PTR	606
6.36.2.10 REGION_SUB_REGION_CREATE_FINISH	607
6.36.2.11 REGION_SUB_REGION_CREATE_START	607
6.36.2.12 REGION_USER_NUMBER_FIND	607
6.36.2.13 REGION_USER_NUMBER_FIND_PTR	608
6.36.2.14 REGIONS_FINALISE	608
6.36.2.15 REGIONS_INITIALISE	608
6.36.3 Variable Documentation	608
6.36.3.1 GLOBAL_REGION	608
6.37 SOLVER_MATRICES_ROUTINES Namespace Reference	609
6.37.1 Detailed Description	609
6.37.2 Function Documentation	610
6.37.2.1 SOLVER_MATRICES_CREATE_FINISH	610
6.37.2.2 SOLVER_MATRICES_CREATE_START	610
6.37.2.3 SOLVER_MATRICES_DESTROY	611
6.37.2.4 SOLVER_MATRICES_FINALISE	611
6.37.2.5 SOLVER_MATRICES_INITIALISE	611

6.37.2.6 SOLVER_MATRICES_LIBRARY_TYPE_SET	612
6.37.2.7 SOLVER_MATRICES_OUTPUT	612
6.37.2.8 SOLVER_MATRICES_STORAGE_TYPE_SET	613
6.37.2.9 SOLVER_MATRIX_FINALISE	613
6.37.2.10 SOLVER_MATRIX_INITIALISE	614
6.37.2.11 SOLVER_MATRIX_STRUCTURE_CALCULATE	614
6.38 SOLVER_ROUTINES Namespace Reference	616
6.38.1 Detailed Description	621
6.38.2 Function Documentation	621
6.38.2.1 SOLVER_CREATE_FINISH	621
6.38.2.2 SOLVER_CREATE_START	621
6.38.2.3 SOLVER_DESTROY	622
6.38.2.4 SOLVER_EIGENPROBLEM_CREATE_FINISH	622
6.38.2.5 SOLVER_EIGENPROBLEM_FINALISE	623
6.38.2.6 SOLVER_EIGENPROBLEM_INITIALISE	623
6.38.2.7 SOLVER_EIGENPROBLEM_SOLVE	624
6.38.2.8 SOLVER_FINALISE	624
6.38.2.9 SOLVER_INITIALISE	624
6.38.2.10 SOLVER_LIBRARY_SET	625
6.38.2.11 SOLVER_LINEAR_CREATE_FINISH	626
6.38.2.12 SOLVER_LINEAR_DIRECT_CREATE_FINISH	626
6.38.2.13 SOLVER_LINEAR_DIRECT_FINALISE	627
6.38.2.14 SOLVER_LINEAR_DIRECT_INITIALISE	627
6.38.2.15 SOLVER_LINEAR_DIRECT_SOLVE	627
6.38.2.16 SOLVER_LINEAR_DIRECT_TYPE_SET	628
6.38.2.17 SOLVER_LINEAR_FINALISE	628
6.38.2.18 SOLVER_LINEAR_INITIALISE	629
6.38.2.19 SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET . .	629
6.38.2.20 SOLVER_LINEAR_ITERATIVE_CREATE_FINISH	630
6.38.2.21 SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET	630
6.38.2.22 SOLVER_LINEAR_ITERATIVE_FINALISE	631
6.38.2.23 SOLVER_LINEAR_ITERATIVE_INITIALISE	631
6.38.2.24 SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET . .	632
6.38.2.25 SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET . .	632
6.38.2.26 SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET . .	633
6.38.2.27 SOLVER_LINEAR_ITERATIVE_SOLVE	633

6.38.2.28 SOLVER_LINEAR_ITERATIVE_TYPE_SET	633
6.38.2.29 SOLVER_LINEAR_SOLVE	634
6.38.2.30 SOLVER_LINEAR_TYPE_SET	634
6.38.2.31 SOLVER_MATRICES_ASSEMBLE	635
6.38.2.32 SOLVER_NONLINEAR_CREATE_FINISH	635
6.38.2.33 SOLVER_NONLINEAR_FINALISE	636
6.38.2.34 SOLVER_NONLINEAR_INITIALISE	636
6.38.2.35 SOLVER_NONLINEAR_SOLVE	637
6.38.2.36 SOLVER_OUTPUT_TYPE_SET	637
6.38.2.37 SOLVER_SOLVE	637
6.38.2.38 SOLVER_SPARSITY_TYPE_SET	638
6.38.2.39 SOLVER_TIME_INTEGRATION_CREATE_FINISH	638
6.38.2.40 SOLVER_TIME_INTEGRATION_FINALISE	639
6.38.2.41 SOLVER_TIME_INTEGRATION_INITIALISE	639
6.38.2.42 SOLVER_TIME_INTEGRATION_SOLVE	640
6.38.2.43 SOLVER_VARIABLES_UPDATE	640
6.38.3 Variable Documentation	641
6.38.3.1 SOLVER_4TH_RUNGE_KUTTA_INTEGRATOR	641
6.38.3.2 SOLVER_ADAMS_MOULTON_INTEGRATOR	641
6.38.3.3 SOLVER_EULER_INTEGRATOR	641
6.38.3.4 SOLVER_IMPROVED_EULER_INTEGRATOR	641
6.38.3.5 SOLVER_LSODA_INTEGRATOR	641
6.39 SORTING Namespace Reference	642
6.39.1 Detailed Description	642
6.39.2 Function Documentation	642
6.39.2.1 BUBBLE_SORT_DP	642
6.39.2.2 BUBBLE_SORT_INTG	642
6.39.2.3 BUBBLE_SORT_SP	643
6.39.2.4 HEAP_SORT_DP	643
6.39.2.5 HEAP_SORT_INTG	643
6.39.2.6 HEAP_SORT_SP	643
6.39.2.7 SHELL_SORT_DP	643
6.39.2.8 SHELL_SORT_INTG	644
6.39.2.9 SHELL_SORT_SP	644
6.40 STRINGS Namespace Reference	645
6.40.1 Detailed Description	649

6.40.2 Function Documentation	649
6.40.2.1 CHARACTER_TO_LOWERCase_C	649
6.40.2.2 CHARACTER_TO_LOWERCase_VS	649
6.40.2.3 CHARACTER_TO_UPPERCase_C	650
6.40.2.4 CHARACTER_TO_UPPERCase_VS	650
6.40.2.5 IS_ABBREVIATION_C_C	650
6.40.2.6 IS_ABBREVIATION_C_VS	650
6.40.2.7 IS_ABBREVIATION_VS_C	651
6.40.2.8 IS_ABBREVIATION_VS_VS	651
6.40.2.9 IS_DIGIT	651
6.40.2.10 IS_LETTER	651
6.40.2.11 IS_LOWERCase	652
6.40.2.12 IS_UPPERCase	652
6.40.2.13 IS_WHITESPACE	652
6.40.2.14 LIST_TO_CHARACTER_C	652
6.40.2.15 LIST_TO_CHARACTER_DP	653
6.40.2.16 LIST_TO_CHARACTER_INTG	653
6.40.2.17 LIST_TO_CHARACTER_L	654
6.40.2.18 LIST_TO_CHARACTER_LINTG	654
6.40.2.19 LIST_TO_CHARACTER_SP	655
6.40.2.20 LOGICAL_TO_CHARACTER	655
6.40.2.21 LOGICAL_TO_VSTRING	656
6.40.2.22 NUMBER_TO_CHARACTER_DP	656
6.40.2.23 NUMBER_TO_CHARACTER_INTG	657
6.40.2.24 NUMBER_TO_CHARACTER_LINTG	657
6.40.2.25 NUMBER_TO_CHARACTER_SP	658
6.40.2.26 NUMBER_TO_VSTRING_DP	658
6.40.2.27 NUMBER_TO_VSTRING_INTG	659
6.40.2.28 NUMBER_TO_VSTRING_LINTG	659
6.40.2.29 NUMBER_TO_VSTRING_SP	659
6.40.2.30 STRING_TO_DOUBLE_C	660
6.40.2.31 STRING_TO_DOUBLE_VS	660
6.40.2.32 STRING_TO_INTEGER_C	661
6.40.2.33 STRING_TO_INTEGER_VS	661
6.40.2.34 STRING_TO_LOGICAL_C	661
6.40.2.35 STRING_TO_LOGICAL_VS	662

6.40.2.36 STRING_TO_LONG_INTEGER_C	662
6.40.2.37 STRING_TO_LONG_INTEGER_VS	662
6.40.2.38 STRING_TO_SINGLE_C	663
6.40.2.39 STRING_TO_SINGLE_VS	663
6.40.2.40 VSTRING_TO_LOWERCASE_C	663
6.40.2.41 VSTRING_TO_LOWERCASE_VS	664
6.40.2.42 VSTRING_TO_UPPERCASE_C	664
6.40.2.43 VSTRING_TO_UPPERCASE_VS	664
6.41 TIMER Namespace Reference	665
6.41.1 Detailed Description	665
6.41.2 Function Documentation	665
6.41.2.1 CPU_TIMER	665
6.41.3 Variable Documentation	665
6.41.3.1 SYSTEM_CPU	665
6.41.3.2 TOTAL_CPU	665
6.41.3.3 USER_CPU	666
6.42 TREES Namespace Reference	667
6.42.1 Detailed Description	668
6.42.2 Function Documentation	669
6.42.2.1 TREE_CREATE_FINISH	669
6.42.2.2 TREE_CREATE_START	669
6.42.2.3 TREE_DESTROY	670
6.42.2.4 TREE_DETACH_AND_DESTROY	670
6.42.2.5 TREE_DETACH_IN_ORDER	670
6.42.2.6 TREE_FINALISE	671
6.42.2.7 TREE_INITIALISE	671
6.42.2.8 TREE_INSERT_TYPE_SET	672
6.42.2.9 TREE_ITEM_DELETE	672
6.42.2.10 TREE_ITEM_INSERT	673
6.42.2.11 TREE_NODE_FINALISE	673
6.42.2.12 TREE_NODE_INITIALISE	674
6.42.2.13 TREE_NODE_KEY_GET	674
6.42.2.14 TREE_NODE_VALUE_GET	675
6.42.2.15 TREE_NODE_VALUE_SET	675
6.42.2.16 TREE_OUTPUT	675
6.42.2.17 TREE_OUTPUT_IN_ORDER	676

6.42.2.18 TREE_PREDECESSOR	676
6.42.2.19 TREE_SEARCH	677
6.42.2.20 TREE_SUCCESSOR	677
6.43 TYPES Namespace Reference	679
6.43.1 Detailed Description	686
7 Class Documentation	687
7.1 BASE_ROUTINES::FLAG_ERROR Interface Reference	687
7.1.1 Detailed Description	687
7.1.2 Member Function Documentation	687
7.1.2.1 FLAG_ERROR_C	687
7.1.2.2 FLAG_ERROR_VS	688
7.2 BASE_ROUTINES::FLAG_WARNING Interface Reference	689
7.2.1 Detailed Description	689
7.2.2 Member Function Documentation	689
7.2.2.1 FLAG_WARNING_C	689
7.2.2.2 FLAG_WARNING_VS	689
7.3 BASE_ROUTINES::interface Interface Reference	690
7.3.1 Detailed Description	690
7.3.2 Member Function Documentation	690
7.3.2.1 CPUTIMER	690
7.4 BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE Struct Reference	691
7.4.1 Detailed Description	691
7.4.2 Member Data Documentation	691
7.4.2.1 NAME	691
7.4.2.2 NEXT_ROUTINE	692
7.4.2.3 NUMBER_OF_INVOCATIONS	692
7.4.2.4 TOTAL_EXCLUSIVE_CPU_TIME	692
7.4.2.5 TOTAL_EXCLUSIVE_SYSTEM_TIME	692
7.4.2.6 TOTAL_INCLUSIVE_CPU_TIME	692
7.4.2.7 TOTAL_INCLUSIVE_SYSTEM_TIME	692
7.5 BASE_ROUTINES::ROUTINE_LIST_TYPE Struct Reference	693
7.5.1 Detailed Description	693
7.5.2 Member Data Documentation	693
7.5.2.1 HEAD	693
7.6 BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE Struct Reference	694
7.6.1 Detailed Description	694

7.6.2	Member Data Documentation	694
7.6.2.1	DIAGNOSTICS	694
7.6.2.2	EXCLUSIVE_CPU_TIME	695
7.6.2.3	EXCLUSIVE_SYSTEM_TIME	695
7.6.2.4	INCLUSIVE_CPU_TIME	695
7.6.2.5	INCLUSIVE_SYSTEM_TIME	695
7.6.2.6	NAME	695
7.6.2.7	PREVIOUS_ROUTINE	695
7.6.2.8	ROUTINE_LIST_ITEM	695
7.6.2.9	TIMING	695
7.7	BASE_ROUTINES::ROUTINE_STACK_TYPE Struct Reference	697
7.7.1	Detailed Description	697
7.7.2	Member Data Documentation	697
7.7.2.1	STACK_POINTER	697
7.8	BINARY_FILE::BINARY_FILE_INFO_TYPE Struct Reference	698
7.8.1	Detailed Description	698
7.8.2	Member Data Documentation	698
7.8.2.1	ACCESS_TYPE	698
7.8.2.2	BINARY_FILE_REVISION	698
7.8.2.3	CHAR_FORMAT	698
7.8.2.4	CHARACTER_SIZE	698
7.8.2.5	DP_FORMAT	699
7.8.2.6	DP_REAL_SIZE	699
7.8.2.7	DPC_REAL_SIZE	699
7.8.2.8	ENDIAN_TYPE	699
7.8.2.9	FILE_NAME	699
7.8.2.10	FILE_NUMBER	699
7.8.2.11	INT_FORMAT	699
7.8.2.12	INTEGER_SIZE	699
7.8.2.13	LINTEGER_SIZE	699
7.8.2.14	LOGICAL_SIZE	699
7.8.2.15	MACHINE_TYPE	699
7.8.2.16	OS_TYPE	700
7.8.2.17	SINTEGER_SIZE	700
7.8.2.18	SP_FORMAT	700
7.8.2.19	SP_REAL_SIZE	700

7.8.2.20 SPC_REAL_SIZE	700
7.9 BINARY_FILE::BINARY_FILE_TYPE Struct Reference	701
7.9.1 Detailed Description	701
7.9.2 Member Data Documentation	701
7.9.2.1 FILE_INFORMATION	701
7.10 BINARY_FILE::BINARY_TAG_TYPE Struct Reference	702
7.10.1 Detailed Description	702
7.10.2 Member Data Documentation	702
7.10.2.1 HEADER	702
7.10.2.2 INDEX	702
7.10.2.3 NUM_BYTES	702
7.10.2.4 NUM_HEADER_BYTES	702
7.10.2.5 NUM_SUBTAGS	702
7.11 BINARY_FILE::interface Interface Reference	703
7.11.1 Detailed Description	703
7.11.2 Member Function Documentation	703
7.11.2.1 BINARYCLOSEFILE	703
7.11.2.2 BINARYOPENFILE	703
7.11.2.3 BINARYSETFILE	703
7.11.2.4 BINARYSKIPFILE	703
7.11.2.5 ISBINARYFILEOPEN	704
7.11.2.6 ISENDBINARYFILE	704
7.12 BINARY_FILE::READ_BINARY_FILE Interface Reference	705
7.12.1 Detailed Description	705
7.12.2 Member Function Documentation	705
7.12.2.1 READ_BINARY_FILE_CHARACTER	705
7.12.2.2 READ_BINARY_FILE_DP	705
7.12.2.3 READ_BINARY_FILE_DP1	705
7.12.2.4 READ_BINARY_FILE_DPC	706
7.12.2.5 READ_BINARY_FILE_DPC1	706
7.12.2.6 READ_BINARY_FILE_INTG	706
7.12.2.7 READ_BINARY_FILE_INTG1	706
7.12.2.8 READ_BINARY_FILE_LINTG	706
7.12.2.9 READ_BINARY_FILE_LINTG1	706
7.12.2.10 READ_BINARY_FILE_LOGICAL	706
7.12.2.11 READ_BINARY_FILE_LOGICAL1	707

7.12.2.12 READ_BINARY_FILE_SINTG	707
7.12.2.13 READ_BINARY_FILE_SINTG1	707
7.12.2.14 READ_BINARY_FILE_SP	707
7.12.2.15 READ_BINARY_FILE_SP1	707
7.12.2.16 READ_BINARY_FILE_SPC	707
7.12.2.17 READ_BINARY_FILE_SPC1	708
7.13 BINARYFILE::WRITE_BINARYFILE Interface Reference	709
7.13.1 Detailed Description	709
7.13.2 Member Function Documentation	709
7.13.2.1 WRITE_BINARYFILE_CHARACTER	709
7.13.2.2 WRITE_BINARYFILE_DP	709
7.13.2.3 WRITE_BINARYFILE_DP1	709
7.13.2.4 WRITE_BINARYFILE_DPC	710
7.13.2.5 WRITE_BINARYFILE_DPC1	710
7.13.2.6 WRITE_BINARYFILE_INTG	710
7.13.2.7 WRITE_BINARYFILE_INTG1	710
7.13.2.8 WRITE_BINARYFILE_LINTG	710
7.13.2.9 WRITE_BINARYFILE_LINTG1	710
7.13.2.10 WRITE_BINARYFILE_LOGICAL	710
7.13.2.11 WRITE_BINARYFILE_LOGICAL1	711
7.13.2.12 WRITE_BINARYFILE_SINTG	711
7.13.2.13 WRITE_BINARYFILE_SINTG1	711
7.13.2.14 WRITE_BINARYFILE_SP	711
7.13.2.15 WRITE_BINARYFILE_SP1	711
7.13.2.16 WRITE_BINARYFILE_SPC	711
7.13.2.17 WRITE_BINARYFILE_SPC1	712
7.14 BLAS::interface Interface Reference	713
7.14.1 Detailed Description	713
7.14.2 Member Function Documentation	713
7.14.2.1 DASUM	713
7.14.2.2 DAXPY	713
7.14.2.3 DCOPY	713
7.14.2.4 DDOT	714
7.14.2.5 DNRM2	714
7.14.2.6 DROT	714
7.14.2.7 DROTG	714

7.14.2.8 DSCAL	714
7.14.2.9 DTRSV	714
7.14.2.10 SASUM	714
7.14.2.11 SAXPY	714
7.14.2.12 SCOPY	715
7.14.2.13 SDOT	715
7.14.2.14 SNRM2	715
7.14.2.15 SROT	715
7.14.2.16 SROTG	715
7.14.2.17 SSCAL	715
7.14.2.18 STRSV	715
7.15 CMISS_PARMETIS::interface Interface Reference	716
7.15.1 Detailed Description	716
7.15.2 Member Function Documentation	716
7.15.2.1 ParMETIS_V3_PartK	716
7.15.2.2 ParMETIS_V3_PartMeshKway	716
7.16 CMISS_PETSC::interface Interface Reference	717
7.16.1 Detailed Description	718
7.16.2 Member Function Documentation	718
7.16.2.1 ISDestroy	718
7.16.2.2 ISLocalToGlobalMappingApply	719
7.16.2.3 ISLocalToGlobalMappingApplyIS	719
7.16.2.4 ISLocalToGlobalMappingCreate	719
7.16.2.5 ISLocalToGlobalMappingDestroy	719
7.16.2.6 KSPCreate	719
7.16.2.7 KSPDestroy	719
7.16.2.8 KSPGetConvergedReason	719
7.16.2.9 KSPGetIterationNumber	719
7.16.2.10 KSPGetPC	719
7.16.2.11 KSPGetResidualNorm	719
7.16.2.12 KSPSetFromOptions	720
7.16.2.13 KSPSetOperators	720
7.16.2.14 KSPSetTolerances	720
7.16.2.15 KSPSetType	720
7.16.2.16 KSPSetUp	720
7.16.2.17 KSPSolve	720

7.16.2.18 MatAssemblyBegin	720
7.16.2.19 MatAssemblyEnd	720
7.16.2.20 MatCreate	720
7.16.2.21 MatCreateMPIAIJ	720
7.16.2.22 MatCreateMPIDense	721
7.16.2.23 MatCreateSeqAIJ	721
7.16.2.24 MatCreateSeqDense	721
7.16.2.25 MatDestroy	721
7.16.2.26 MatGetArray	721
7.16.2.27 MatGetArrayF90	721
7.16.2.28 MatGetOwnershipRange	721
7.16.2.29 MatGetValues	721
7.16.2.30 MatRestoreArray	721
7.16.2.31 MatRestoreArrayF90	721
7.16.2.32 MatSetLocalToGlobalMapping	722
7.16.2.33 MatSetOption	722
7.16.2.34 MatSetSizes	722
7.16.2.35 MatSetValue	722
7.16.2.36 MatSetValueLocal	722
7.16.2.37 MatSetValues	722
7.16.2.38 MatSetValuesLocal	722
7.16.2.39 MatView	722
7.16.2.40 MatZeroEntries	722
7.16.2.41 PCSetType	722
7.16.2.42 PetscFinalize	723
7.16.2.43 PetscInitialize	723
7.16.2.44 PetscLogPrintSummary	723
7.16.2.45 SNESCreate	723
7.16.2.46 SNESDestroy	723
7.16.2.47 SNESSetFromOptions	723
7.16.2.48 SNESSetFunction	723
7.16.2.49 SNESSetType	723
7.16.2.50 SNESolve	723
7.16.2.51 VecAssemblyBegin	723
7.16.2.52 VecAssemblyEnd	723
7.16.2.53 VecCreate	724

7.16.2.54 VecCreateGhost	724
7.16.2.55 VecCreateGhostWithArray	724
7.16.2.56 VecCreateMPI	724
7.16.2.57 VecCreateMPIWithArray	724
7.16.2.58 VecCreateSeq	724
7.16.2.59 VecCreateSeqWithArray	724
7.16.2.60 VecDestroy	724
7.16.2.61 VecDuplicate	724
7.16.2.62 VecGetArray	724
7.16.2.63 VecGetArrayF90	725
7.16.2.64 VecGetLocalSize	725
7.16.2.65 VecGetOwnershipRange	725
7.16.2.66 VecGetSize	725
7.16.2.67 VecGetValues	725
7.16.2.68 VecGhostGetLocalForm	725
7.16.2.69 VecGhostRestoreLocalForm	725
7.16.2.70 VecGhostUpdateBegin	725
7.16.2.71 VecGhostUpdateEnd	725
7.16.2.72 VecRestoreArray	725
7.16.2.73 VecRestoreArrayF90	726
7.16.2.74 VecSet	726
7.16.2.75 VecSetFromOptions	726
7.16.2.76 VecSetLocalToGlobalMapping	726
7.16.2.77 VecSetSizes	726
7.16.2.78 VecSetValue	726
7.16.2.79 VecSetValuesLocal	726
7.16.2.80 VecView	726
7.17 CMISS_PETSC_TYPES::PETSC_IS_TYPE Struct Reference	727
7.17.1 Detailed Description	727
7.18 CMISS_PETSC_TYPES::PETSC_ISLOCALTOGLOBALMAPPING_TYPE Struct Reference	728
7.18.1 Detailed Description	728
7.19 CMISS_PETSC_TYPES::PETSC_KSP_TYPE Struct Reference	729
7.19.1 Detailed Description	729
7.20 CMISS_PETSC_TYPES::PETSC_MAT_TYPE Struct Reference	730
7.20.1 Detailed Description	730
7.21 CMISS_PETSC_TYPES::PETSC_PC_TYPE Struct Reference	731

7.21.1	Detailed Description	731
7.22	CMISS_PETSC_TYPES::PETSC_SNES_TYPE Struct Reference	732
7.22.1	Detailed Description	732
7.23	CMISS_PETSC_TYPES::PETSC_VEC_TYPE Struct Reference	733
7.23.1	Detailed Description	733
7.24	COMP_ENVIRONMENT::CACHE_TYPE Struct Reference	734
7.24.1	Detailed Description	734
7.24.2	Member Data Documentation	734
7.24.2.1	NUMBER_LEVELS	734
7.24.2.2	SIZE	734
7.25	COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE Struct Reference	735
7.25.1	Detailed Description	735
7.25.2	Member Data Documentation	735
7.25.2.1	COMPUTATIONAL_NODES	735
7.25.2.2	MPI_COMM	735
7.25.2.3	MY_COMPUTATIONAL_NODE_NUMBER	735
7.25.2.4	NUMBER_COMPUTATIONAL_NODES	736
7.26	COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE Struct Reference	737
7.26.1	Detailed Description	737
7.26.2	Member Data Documentation	737
7.26.2.1	NODE_NAME	737
7.26.2.2	NODE_NAME_LENGTH	737
7.26.2.3	NUMBER_PROCESSORS	737
7.26.2.4	RANK	738
7.27	COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE Struct Reference	739
7.27.1	Detailed Description	739
7.27.2	Member Data Documentation	739
7.27.2.1	BLOCK_LENGTHS	739
7.27.2.2	DISPLACEMENTS	739
7.27.2.3	MPI_TYPE	739
7.27.2.4	NUM_BLOCKS	740
7.27.2.5	TYPES	740
7.28	COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC Interface Reference	741
7.28.1	Detailed Description	741
7.28.2	Member Function Documentation	741
7.28.2.1	COORDINATE_CONVERT_FROM_RC_DP	741

7.28.2.2	COORDINATE_CONVERT_FROM_RC_SP	741
7.29	COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC Interface Reference	742
7.29.1	Detailed Description	742
7.29.2	Member Function Documentation	742
7.29.2.1	COORDINATE_CONVERT_TO_RC_DP	742
7.29.2.2	COORDINATE_CONVERT_TO_RC_SP	742
7.30	COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE Interface Reference	743
7.30.1	Detailed Description	743
7.30.2	Member Function Documentation	743
7.30.2.1	COORDINATE_DELTA_CALCULATE_DP	743
7.31	COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_TO_RC Interface Reference	744
7.31.1	Detailed Description	744
7.31.2	Member Function Documentation	744
7.31.2.1	COORDINATE_DERIVATIVE_CONVERT_TO_RC_DP	744
7.31.2.2	COORDINATE_DERIVATIVE_CONVERT_TO_RC_SP	744
7.32	COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY Interface Reference	745
7.32.1	Detailed Description	745
7.32.2	Member Function Documentation	745
7.32.2.1	COORDINATE_SYSTEM_DESTROY_NUMBER	745
7.32.2.2	COORDINATE_SYSTEM_DESTROY_PTR	745
7.33	COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET Interface Reference	746
7.33.1	Detailed Description	746
7.33.2	Member Function Documentation	746
7.33.2.1	COORDINATE_SYSTEM_DIMENSION_SET_NUMBER	746
7.33.2.2	COORDINATE_SYSTEM_DIMENSION_SET_PTR	746
7.34	COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET Interface Reference	747
7.34.1	Detailed Description	747
7.34.2	Member Function Documentation	747
7.34.2.1	COORDINATE_SYSTEM_FOCUS_SET_NUMBER	747
7.34.2.2	COORDINATE_SYSTEM_FOCUS_SET_PTR	747
7.35	COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET Interface Reference	748
7.35.1	Detailed Description	748
7.35.2	Member Function Documentation	748
7.35.2.1	COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER	748

7.35.2.2 COORDINATE_SYSTEM_ORIENTATION_SET_PTR	748
7.36 COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET Interface Reference	749
7.36.1 Detailed Description	749
7.36.2 Member Function Documentation	749
7.36.2.1 COORDINATE_SYSTEM_ORIGIN_SET_NUMBER	749
7.36.2.2 COORDINATE_SYSTEM_ORIGIN_SET_PTR	749
7.37 COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE Struct Reference	750
7.37.1 Detailed Description	750
7.37.2 Member Data Documentation	750
7.37.2.1 PTR	750
7.38 COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_- INTERPOLATION_TYPE_SET Interface Reference	751
7.38.1 Detailed Description	751
7.38.2 Member Function Documentation	751
7.38.2.1 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_NUMBER	751
7.38.2.2 COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_- SET_PTR	751
7.39 COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET Interface Reference	752
7.39.1 Detailed Description	752
7.39.2 Member Function Documentation	752
7.39.2.1 COORDINATE_SYSTEM_TYPE_SET_NUMBER	752
7.39.2.2 COORDINATE_SYSTEM_TYPE_SET_PTR	752
7.40 COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE Struct Reference	753
7.40.1 Detailed Description	753
7.40.2 Member Data Documentation	753
7.40.2.1 COORDINATE_SYSTEMS	753
7.40.2.2 NUMBER_OF_COORDINATE_SYSTEMS	753
7.41 COORDINATE_ROUTINES::D2ZX Interface Reference	754
7.41.1 Detailed Description	754
7.41.2 Member Function Documentation	754
7.41.2.1 D2ZX_DP	754
7.42 COORDINATE_ROUTINES::DXZ Interface Reference	755
7.42.1 Detailed Description	755
7.42.2 Member Function Documentation	755
7.42.2.1 DXZ_DP	755
7.43 COORDINATE_ROUTINES::DZX Interface Reference	756

7.43.1	Detailed Description	756
7.43.2	Member Function Documentation	756
7.43.2.1	DZX_DP	756
7.44	EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_- DOF Interface Reference	757
7.44.1	Detailed Description	757
7.44.2	Member Function Documentation	757
7.44.2.1	EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1	757
7.44.2.2	EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS	757
7.45	EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICATION_SET Interface Reference	759
7.45.1	Detailed Description	759
7.45.2	Member Function Documentation	759
7.45.2.1	EQUATIONS_SET_SPECIFICATION_SET_NUMBER	759
7.45.2.2	EQUATIONS_SET_SPECIFICATION_SET_PTR	759
7.46	F90C::interface Interface Reference	760
7.46.1	Detailed Description	760
7.46.2	Member Function Documentation	760
7.46.2.1	CSTRINGLEN	760
7.46.2.2	PACKCHARACTERS	760
7.46.2.3	UNPACKCHARACTERS	760
7.47	FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET Struct Reference	761
7.47.1	Detailed Description	761
7.47.2	Member Data Documentation	761
7.47.2.1	ELEMENTAL_INFO_SET	761
7.47.2.2	FIELDS	761
7.47.2.3	LIST_OF_GLOBAL_NUMBER	761
7.47.2.4	NUMBER_OF_ELEMENTS	762
7.48	FIELD_IO_ROUTINES::FIELD_IO_INFO_SET Struct Reference	763
7.48.1	Detailed Description	763
7.48.2	Member Data Documentation	763
7.48.2.1	COMPONENTS	763
7.48.2.2	NUMBER_OF_COMPONENTS	763
7.48.2.3	SAME_HEADER	763
7.49	FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET Struct Reference	764
7.49.1	Detailed Description	764
7.49.2	Member Data Documentation	764

7.49.2.1 FIELDS	764
7.49.2.2 LIST_OF_GLOBAL_NUMBER	764
7.49.2.3 NODAL_INFO_SET	764
7.49.2.4 NUMBER_OF_NODES	765
7.50 FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE Struct Reference	766
7.50.1 Detailed Description	766
7.50.2 Member Data Documentation	766
7.50.2.1 PTR	766
7.51 FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE Struct Reference	767
7.51.1 Detailed Description	767
7.51.2 Member Data Documentation	767
7.51.2.1 PTR	767
7.52 FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_SET Interface Reference	768
7.52.1 Detailed Description	768
7.52.2 Member Function Documentation	768
7.52.2.1 FIELD_COMPONENT_INTERPOLATION_SET_NUMBER	768
7.52.2.2 FIELD_COMPONENT_INTERPOLATION_SET_PTR	768
7.53 FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET Interface Reference	769
7.53.1 Detailed Description	769
7.53.2 Member Function Documentation	769
7.53.2.1 FIELD_COMPONENT_MESH_COMPONENT_SET_NUMBER	769
7.53.2.2 FIELD_COMPONENT_MESH_COMPONENT_SET_PTR	769
7.54 FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET Interface Reference	770
7.54.1 Detailed Description	770
7.54.2 Member Function Documentation	770
7.54.2.1 FIELD_DEPENDENT_TYPE_SET_NUMBER	770
7.54.2.2 FIELD_DEPENDENT_TYPE_SET_PTR	770
7.55 FIELD_ROUTINES::FIELD_DIMENSION_SET Interface Reference	771
7.55.1 Detailed Description	771
7.55.2 Member Function Documentation	771
7.55.2.1 FIELD_DIMENSION_SET_NUMBER	771
7.55.2.2 FIELD_DIMENSION_SET_PTR	771
7.56 FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET Interface Reference	772
7.56.1 Detailed Description	772
7.56.2 Member Function Documentation	772

7.56.2.1	FIELD_GEOMETRIC_FIELD_SET_NUMBER	772
7.56.2.2	FIELD_GEOMETRIC_FIELD_SET_PTR	772
7.57	FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET Interface Reference	773
7.57.1	Detailed Description	773
7.57.2	Member Function Documentation	773
7.57.2.1	FIELD_MESH_DECOMPOSITION_SET_NUMBER	773
7.57.2.2	FIELD_MESH_DECOMPOSITION_SET_PTR	773
7.58	FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET Interface Reference	774
7.58.1	Detailed Description	774
7.58.2	Member Function Documentation	774
7.58.2.1	FIELD_NUMBER_OF_COMPONENTS_SET_NUMBER	774
7.58.2.2	FIELD_NUMBER_OF_COMPONENTS_SET_PTR	774
7.59	FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET Interface Reference	775
7.59.1	Detailed Description	775
7.59.2	Member Function Documentation	775
7.59.2.1	FIELD_NUMBER_OF_VARIABLES_SET_NUMBER	775
7.59.2.2	FIELD_NUMBER_OF_VARIABLES_SET_PTR	775
7.60	FIELD_ROUTINES::FIELD_SCALING_TYPE_SET Interface Reference	776
7.60.1	Detailed Description	776
7.60.2	Member Function Documentation	776
7.60.2.1	FIELD_SCALING_TYPE_SET_NUMBER	776
7.60.2.2	FIELD_SCALING_TYPE_SET_PTR	776
7.61	FIELD_ROUTINES::FIELD_TYPE_SET Interface Reference	777
7.61.1	Detailed Description	777
7.61.2	Member Function Documentation	777
7.61.2.1	FIELD_TYPE_SET_NUMBER	777
7.61.2.2	FIELD_TYPE_SET_PTR	777
7.62	INPUT_OUTPUT::WRITE_STRING Interface Reference	778
7.62.1	Detailed Description	778
7.62.2	Member Function Documentation	778
7.62.2.1	WRITE_STRING_C	778
7.62.2.2	WRITE_STRING_VS	778
7.63	INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE Interface Reference	779
7.63.1	Detailed Description	779
7.63.2	Member Function Documentation	781
7.63.2.1	WRITE_STRING_FMT_TWO_VALUE_C_C	781

7.63.2.2	WRITE_STRING_FMT_TWO_VALUE_C_DP	781
7.63.2.3	WRITE_STRING_FMT_TWO_VALUE_C_INTG	781
7.63.2.4	WRITE_STRING_FMT_TWO_VALUE_C_L	781
7.63.2.5	WRITE_STRING_FMT_TWO_VALUE_C_SP	781
7.63.2.6	WRITE_STRING_FMT_TWO_VALUE_C_VS	781
7.63.2.7	WRITE_STRING_FMT_TWO_VALUE_DP_C	781
7.63.2.8	WRITE_STRING_FMT_TWO_VALUE_DP_DP	781
7.63.2.9	WRITE_STRING_FMT_TWO_VALUE_DP_INTG	781
7.63.2.10	WRITE_STRING_FMT_TWO_VALUE_DP_L	781
7.63.2.11	WRITE_STRING_FMT_TWO_VALUE_DP_SP	781
7.63.2.12	WRITE_STRING_FMT_TWO_VALUE_DP_VS	781
7.63.2.13	WRITE_STRING_FMT_TWO_VALUE_INTG_C	781
7.63.2.14	WRITE_STRING_FMT_TWO_VALUE_INTG_DP	781
7.63.2.15	WRITE_STRING_FMT_TWO_VALUE_INTG_INTG	781
7.63.2.16	WRITE_STRING_FMT_TWO_VALUE_INTG_L	781
7.63.2.17	WRITE_STRING_FMT_TWO_VALUE_INTG_SP	781
7.63.2.18	WRITE_STRING_FMT_TWO_VALUE_INTG_VS	781
7.63.2.19	WRITE_STRING_FMT_TWO_VALUE_L_C	781
7.63.2.20	WRITE_STRING_FMT_TWO_VALUE_L_DP	781
7.63.2.21	WRITE_STRING_FMT_TWO_VALUE_L_INTG	781
7.63.2.22	WRITE_STRING_FMT_TWO_VALUE_L_L	781
7.63.2.23	WRITE_STRING_FMT_TWO_VALUE_L_SP	781
7.63.2.24	WRITE_STRING_FMT_TWO_VALUE_L_VS	781
7.63.2.25	WRITE_STRING_FMT_TWO_VALUE_SP_C	781
7.63.2.26	WRITE_STRING_FMT_TWO_VALUE_SP_DP	781
7.63.2.27	WRITE_STRING_FMT_TWO_VALUE_SP_INTG	781
7.63.2.28	WRITE_STRING_FMT_TWO_VALUE_SP_L	781
7.63.2.29	WRITE_STRING_FMT_TWO_VALUE_SP_SP	781
7.63.2.30	WRITE_STRING_FMT_TWO_VALUE_SP_VS	781
7.63.2.31	WRITE_STRING_FMT_TWO_VALUE_VS_C	781
7.63.2.32	WRITE_STRING_FMT_TWO_VALUE_VS_DP	781
7.63.2.33	WRITE_STRING_FMT_TWO_VALUE_VS_INTG	781
7.63.2.34	WRITE_STRING_FMT_TWO_VALUE_VS_L	781
7.63.2.35	WRITE_STRING_FMT_TWO_VALUE_VS_SP	781
7.63.2.36	WRITE_STRING_FMT_TWO_VALUE_VS_VS	781
7.64	INPUT_OUTPUT::WRITE_STRING_FMT_VALUE Interface Reference	782

7.64.1	Detailed Description	782
7.64.2	Member Function Documentation	782
7.64.2.1	WRITE_STRING_FMT_VALUE_C	782
7.64.2.2	WRITE_STRING_FMT_VALUE_DP	783
7.64.2.3	WRITE_STRING_FMT_VALUE_INTG	783
7.64.2.4	WRITE_STRING_FMT_VALUE_L	783
7.64.2.5	WRITE_STRING_FMT_VALUE_LINTG	784
7.64.2.6	WRITE_STRING_FMT_VALUE_SP	784
7.64.2.7	WRITE_STRING_FMT_VALUE_VS	785
7.65	INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR Interface Reference	786
7.65.1	Detailed Description	786
7.65.2	Member Function Documentation	786
7.65.2.1	WRITE_STRING_IDX_VECTOR_DP	786
7.65.2.2	WRITE_STRING_IDX_VECTOR_INTG	786
7.65.2.3	WRITE_STRING_IDX_VECTOR_L	786
7.65.2.4	WRITE_STRING_IDX_VECTOR_LINTG	786
7.65.2.5	WRITE_STRING_IDX_VECTOR_SP	786
7.66	INPUT_OUTPUT::WRITE_STRING_MATRIX Interface Reference	787
7.66.1	Detailed Description	787
7.66.2	Member Function Documentation	787
7.66.2.1	WRITE_STRING_MATRIX_DP	787
7.66.2.2	WRITE_STRING_MATRIX_INTG	788
7.66.2.3	WRITE_STRING_MATRIX_L	789
7.66.2.4	WRITE_STRING_MATRIX_LINTG	790
7.66.2.5	WRITE_STRING_MATRIX_SP	791
7.67	INPUT_OUTPUT::WRITE_STRING_TWO_VALUE Interface Reference	792
7.67.1	Detailed Description	793
7.67.2	Member Function Documentation	793
7.67.2.1	WRITE_STRING_TWO_VALUE_C_C	793
7.67.2.2	WRITE_STRING_TWO_VALUE_C_DP	794
7.67.2.3	WRITE_STRING_TWO_VALUE_C_INTG	794
7.67.2.4	WRITE_STRING_TWO_VALUE_C_L	795
7.67.2.5	WRITE_STRING_TWO_VALUE_C_SP	795
7.67.2.6	WRITE_STRING_TWO_VALUE_C_VS	796
7.67.2.7	WRITE_STRING_TWO_VALUE_DP_C	796
7.67.2.8	WRITE_STRING_TWO_VALUE_DP_DP	797

7.67.2.9	WRITE_STRING_TWO_VALUE_DP_INTG	797
7.67.2.10	WRITE_STRING_TWO_VALUE_DP_L	798
7.67.2.11	WRITE_STRING_TWO_VALUE_DP_SP	798
7.67.2.12	WRITE_STRING_TWO_VALUE_DP_VS	799
7.67.2.13	WRITE_STRING_TWO_VALUE_INTG_C	799
7.67.2.14	WRITE_STRING_TWO_VALUE_INTG_DP	800
7.67.2.15	WRITE_STRING_TWO_VALUE_INTG_INTG	800
7.67.2.16	WRITE_STRING_TWO_VALUE_INTG_L	801
7.67.2.17	WRITE_STRING_TWO_VALUE_INTG_SP	801
7.67.2.18	WRITE_STRING_TWO_VALUE_INTG_VS	802
7.67.2.19	WRITE_STRING_TWO_VALUE_L_C	802
7.67.2.20	WRITE_STRING_TWO_VALUE_L_DP	803
7.67.2.21	WRITE_STRING_TWO_VALUE_L_INTG	803
7.67.2.22	WRITE_STRING_TWO_VALUE_L_L	804
7.67.2.23	WRITE_STRING_TWO_VALUE_L_SP	804
7.67.2.24	WRITE_STRING_TWO_VALUE_L_VS	805
7.67.2.25	WRITE_STRING_TWO_VALUE_SP_C	805
7.67.2.26	WRITE_STRING_TWO_VALUE_SP_DP	806
7.67.2.27	WRITE_STRING_TWO_VALUE_SP_INTG	806
7.67.2.28	WRITE_STRING_TWO_VALUE_SP_L	807
7.67.2.29	WRITE_STRING_TWO_VALUE_SP_SP	807
7.67.2.30	WRITE_STRING_TWO_VALUE_SP_VS	808
7.67.2.31	WRITE_STRING_TWO_VALUE_VS_C	808
7.67.2.32	WRITE_STRING_TWO_VALUE_VS_DP	809
7.67.2.33	WRITE_STRING_TWO_VALUE_VS_INTG	809
7.67.2.34	WRITE_STRING_TWO_VALUE_VS_L	810
7.67.2.35	WRITE_STRING_TWO_VALUE_VS_SP	810
7.67.2.36	WRITE_STRING_TWO_VALUE_VS_VS	811
7.68	INPUT_OUTPUT::WRITE_STRING_VALUE Interface Reference	812
7.68.1	Detailed Description	812
7.68.2	Member Function Documentation	812
7.68.2.1	WRITE_STRING_VALUE_C	812
7.68.2.2	WRITE_STRING_VALUE_DP	812
7.68.2.3	WRITE_STRING_VALUE_INTG	813
7.68.2.4	WRITE_STRING_VALUE_L	813
7.68.2.5	WRITE_STRING_VALUE_LINTG	814

7.68.2.6	WRITE_STRING_VALUE_SP	814
7.68.2.7	WRITE_STRING_VALUE_VS	814
7.69	INPUT_OUTPUT::WRITE_STRING_VECTOR Interface Reference	815
7.69.1	Detailed Description	815
7.69.2	Member Function Documentation	815
7.69.2.1	WRITE_STRING_VECTOR_DP	815
7.69.2.2	WRITE_STRING_VECTOR_INTG	815
7.69.2.3	WRITE_STRING_VECTOR_L	815
7.69.2.4	WRITE_STRING_VECTOR_LINTG	815
7.69.2.5	WRITE_STRING_VECTOR_SP	815
7.70	ISO_VARYING_STRING::adjustr Interface Reference	816
7.70.1	Detailed Description	816
7.70.2	Member Function Documentation	816
7.70.2.1	adjustr_	816
7.71	ISO_VARYING_STRING::assignment(=) Interface Reference	817
7.71.1	Detailed Description	817
7.71.2	Member Function Documentation	817
7.71.2.1	adjustl_	817
7.71.2.2	op_assign_CH_VS	817
7.71.2.3	op_assign_VS_CH	817
7.71.2.4	op_concat_CH_VS	818
7.71.2.5	op_concat_VS_CH	818
7.71.2.6	op_concat_VS_VS	818
7.71.2.7	op_eq_CH_VS	818
7.71.2.8	op_eq_VS_CH	818
7.71.2.9	op_eq_VS_VS	818
7.71.2.10	op_ge_CH_VS	818
7.71.2.11	op_ge_VS_CH	818
7.71.2.12	op_ge_VS_VS	818
7.71.2.13	op_gt_CH_VS	819
7.71.2.14	op_gt_VS_CH	819
7.71.2.15	op_gt_VS_VS	819
7.71.2.16	op_le_CH_VS	819
7.71.2.17	op_le_VS_CH	819
7.71.2.18	op_le_VS_VS	819
7.71.2.19	op_lt_CH_VS	819

7.71.2.20 op_lt_VS_CH	819
7.71.2.21 op_lt_VS_VS	819
7.71.2.22 op_ne_CH_VS	820
7.71.2.23 op_ne_VS_CH	820
7.71.2.24 op_ne_VS_VS	820
7.72 ISO_VARYING_STRING::char Interface Reference	821
7.72.1 Detailed Description	821
7.72.2 Member Function Documentation	821
7.72.2.1 char_auto	821
7.72.2.2 char_fixed	821
7.73 ISO_VARYING_STRING::extract Interface Reference	822
7.73.1 Detailed Description	822
7.73.2 Member Function Documentation	822
7.73.2.1 extract_CH	822
7.73.2.2 extract_VS	822
7.74 ISO_VARYING_STRING::get Interface Reference	823
7.74.1 Detailed Description	823
7.74.2 Member Function Documentation	823
7.74.2.1 get_	823
7.74.2.2 get_set_CH	823
7.74.2.3 get_set_VS	823
7.74.2.4 get_unit	823
7.74.2.5 get_unit_set_CH	823
7.74.2.6 get_unit_set_VS	824
7.75 ISO_VARYING_STRING::iachar Interface Reference	825
7.75.1 Detailed Description	825
7.75.2 Member Function Documentation	825
7.75.2.1 iachar_	825
7.76 ISO_VARYING_STRING::ichar Interface Reference	826
7.76.1 Detailed Description	826
7.76.2 Member Function Documentation	826
7.76.2.1 ichar_	826
7.77 ISO_VARYING_STRING::index Interface Reference	827
7.77.1 Detailed Description	827
7.77.2 Member Function Documentation	827
7.77.2.1 index_CH_VS	827

7.77.2.2	index_VS_CH	827
7.77.2.3	index_VS_VS	827
7.78	ISO_VARYING_STRING::insert Interface Reference	828
7.78.1	Detailed Description	828
7.78.2	Member Function Documentation	828
7.78.2.1	insert_CH_CH	828
7.78.2.2	insert_CH_VS	828
7.78.2.3	insert_VS_CH	828
7.78.2.4	insert_VS_VS	828
7.79	ISO_VARYING_STRING::len Interface Reference	829
7.79.1	Detailed Description	829
7.79.2	Member Function Documentation	829
7.79.2.1	len_	829
7.80	ISO_VARYING_STRING::len_trim Interface Reference	830
7.80.1	Detailed Description	830
7.80.2	Member Function Documentation	830
7.80.2.1	len_trim_	830
7.81	ISO_VARYING_STRING::lge Interface Reference	831
7.81.1	Detailed Description	831
7.81.2	Member Function Documentation	831
7.81.2.1	lge_CH_VS	831
7.81.2.2	lge_VS_CH	831
7.81.2.3	lge_VS_VS	831
7.82	ISO_VARYING_STRING::lgt Interface Reference	832
7.82.1	Detailed Description	832
7.82.2	Member Function Documentation	832
7.82.2.1	lgt_CH_VS	832
7.82.2.2	lgt_VS_CH	832
7.82.2.3	lgt_VS_VS	832
7.83	ISO_VARYING_STRING::lle Interface Reference	833
7.83.1	Detailed Description	833
7.83.2	Member Function Documentation	833
7.83.2.1	lle_CH_VS	833
7.83.2.2	lle_VS_CH	833
7.83.2.3	lle_VS_VS	833
7.84	ISO_VARYING_STRING::llt Interface Reference	834

7.84.1	Detailed Description	834
7.84.2	Member Function Documentation	834
7.84.2.1	llt_CH_VS	834
7.84.2.2	llt_VS_CH	834
7.84.2.3	llt_VS_VS	834
7.85	ISO_VARYING_STRING::put Interface Reference	835
7.85.1	Detailed Description	835
7.85.2	Member Function Documentation	835
7.85.2.1	put_CH	835
7.85.2.2	put_unit_CH	835
7.85.2.3	put_unit_VS	835
7.85.2.4	put_VS	835
7.86	ISO_VARYING_STRING::put_line Interface Reference	836
7.86.1	Detailed Description	836
7.86.2	Member Function Documentation	836
7.86.2.1	put_line_CH	836
7.86.2.2	put_line_unit_CH	836
7.86.2.3	put_line_unit_VS	836
7.86.2.4	put_line_VS	836
7.87	ISO_VARYING_STRING::remove Interface Reference	837
7.87.1	Detailed Description	837
7.87.2	Member Function Documentation	837
7.87.2.1	remove_CH	837
7.87.2.2	remove_VS	837
7.88	ISO_VARYING_STRING::repeat Interface Reference	838
7.88.1	Detailed Description	838
7.88.2	Member Function Documentation	838
7.88.2.1	repeat_	838
7.89	ISO_VARYING_STRING::replace Interface Reference	839
7.89.1	Detailed Description	839
7.89.2	Member Function Documentation	839
7.89.2.1	replace_CH_CH_auto	839
7.89.2.2	replace_CH_CH_CH_target	839
7.89.2.3	replace_CH_CH_fixed	839
7.89.2.4	replace_CH_CH_VS_target	840
7.89.2.5	replace_CH_VS_auto	840

7.89.2.6 replace_CH_VS_CH_target	840
7.89.2.7 replace_CH_VS_fixed	840
7.89.2.8 replace_CH_VS_VS_target	840
7.89.2.9 replace_VS_CH_auto	840
7.89.2.10 replace_VS_CH_CH_target	840
7.89.2.11 replace_VS_CH_fixed	841
7.89.2.12 replace_VS_CH_VS_target	841
7.89.2.13 replace_VS_VS_auto	841
7.89.2.14 replace_VS_VS_CH_target	841
7.89.2.15 replace_VS_VS_fixed	841
7.89.2.16 replace_VS_VS_VS_target	841
7.90 ISO_VARYING_STRING::scan Interface Reference	842
7.90.1 Detailed Description	842
7.90.2 Member Function Documentation	842
7.90.2.1 scan_CH_VS	842
7.90.2.2 scan_VS_CH	842
7.90.2.3 scan_VS_VS	842
7.91 ISO_VARYING_STRING::split Interface Reference	843
7.91.1 Detailed Description	843
7.91.2 Member Function Documentation	843
7.91.2.1 split_CH	843
7.91.2.2 split_VS	843
7.92 ISO_VARYING_STRING::trim Interface Reference	844
7.92.1 Detailed Description	844
7.92.2 Member Function Documentation	844
7.92.2.1 trim_	844
7.93 ISO_VARYING_STRING::var_str Interface Reference	845
7.93.1 Detailed Description	845
7.93.2 Member Function Documentation	845
7.93.2.1 var_str_	845
7.94 ISO_VARYING_STRING::VARYING_STRING Struct Reference	846
7.94.1 Detailed Description	846
7.94.2 Member Data Documentation	846
7.94.2.1 chars	846
7.95 ISO_VARYING_STRING::verify Interface Reference	847
7.95.1 Detailed Description	847

7.95.2 Member Function Documentation	847
7.95.2.1 verify_CH_VS	847
7.95.2.2 verify_VS_CH	847
7.95.2.3 verify_VS_VS	847
7.96 LAPACK::interface Interface Reference	848
7.96.1 Detailed Description	848
7.96.2 Member Function Documentation	848
7.96.2.1 DGESV	848
7.97 LISTS::LIST_DETACH_AND_DESTROY Interface Reference	849
7.97.1 Detailed Description	849
7.97.2 Member Function Documentation	849
7.97.2.1 LIST_DETACH_AND_DESTROY_DP	849
7.97.2.2 LIST_DETACH_AND_DESTROY_INTG	850
7.97.2.3 LIST_DETACH_AND_DESTROY_SP	850
7.98 LISTS::LIST_ITEM_ADD Interface Reference	851
7.98.1 Detailed Description	851
7.98.2 Member Function Documentation	851
7.98.2.1 LIST_ITEM_ADD_DP1	851
7.98.2.2 LIST_ITEM_ADD_INTG1	851
7.98.2.3 LIST_ITEM_ADD_SP1	852
7.99 LISTS::LIST_ITEM_IN_LIST Interface Reference	853
7.99.1 Detailed Description	853
7.99.2 Member Function Documentation	853
7.99.2.1 LIST_ITEM_IN_LIST_DP1	853
7.99.2.2 LIST_ITEM_IN_LIST_INTG1	853
7.99.2.3 LIST_ITEM_IN_LIST_SP1	854
7.100 LISTS::LIST_PTR_TYPE Struct Reference	855
7.100.1 Detailed Description	855
7.100.2 Member Data Documentation	855
7.100.2.1 PTR	855
7.101 LISTS::LIST_SEARCH Interface Reference	856
7.101.1 Detailed Description	856
7.101.2 Member Function Documentation	856
7.101.2.1 LIST_SEARCH_DP_ARRAY	856
7.101.2.2 LIST_SEARCH_INTG_ARRAY	856
7.101.2.3 LIST_SEARCH_SP_ARRAY	857

7.102LISTS::LIST_SEARCH_LINEAR Interface Reference	858
7.102.1 Detailed Description	858
7.102.2 Member Function Documentation	858
7.102.2.1 LIST_SEARCH_LINEAR_DP_ARRAY	858
7.102.2.2 LIST_SEARCH_LINEAR_INTG_ARRAY	858
7.102.2.3 LIST_SEARCH_LINEAR_SP_ARRAY	859
7.103LISTS::LIST_SORT Interface Reference	860
7.103.1 Detailed Description	860
7.103.2 Member Function Documentation	860
7.103.2.1 LIST_SORT_DP_ARRAY	860
7.103.2.2 LIST_SORT_INTG_ARRAY	860
7.103.2.3 LIST_SORT_SP_ARRAY	860
7.104LISTS::LIST_SORT_BUBBLE Interface Reference	862
7.104.1 Detailed Description	862
7.104.2 Member Function Documentation	862
7.104.2.1 LIST_SORT_BUBBLE_DP_ARRAY	862
7.104.2.2 LIST_SORT_BUBBLE_INTG_ARRAY	862
7.104.2.3 LIST_SORT_BUBBLE_SP_ARRAY	862
7.105LISTS::LIST_SORT_HEAP Interface Reference	864
7.105.1 Detailed Description	864
7.105.2 Member Function Documentation	864
7.105.2.1 LIST_SORT_HEAP_DP_ARRAY	864
7.105.2.2 LIST_SORT_HEAP_INTG_ARRAY	864
7.105.2.3 LIST_SORT_HEAP_SP_ARRAY	864
7.106LISTS::LIST_SORT_SHELL Interface Reference	866
7.106.1 Detailed Description	866
7.106.2 Member Function Documentation	866
7.106.2.1 LIST_SORT_SHELL_DP_ARRAY	866
7.106.2.2 LIST_SORT_SHELL_INTG_ARRAY	866
7.106.2.3 LIST_SORT_SHELL_SP_ARRAY	866
7.107LISTS::LIST_TYPE Struct Reference	867
7.107.1 Detailed Description	867
7.107.2 Member Data Documentation	867
7.107.2.1 DATA_TYPE	867
7.107.2.2 INITIAL_SIZE	868
7.107.2.3 LIST_DP	868

7.107.2.4 LIST_FINISHED	868
7.107.2.5 LIST_INTG	868
7.107.2.6 LIST_SP	868
7.107.2.7 NUMBER_IN_LIST	868
7.107.2.8 SIZE	868
7.107.2.9 SORT_METHOD	868
7.107.2.10 SORT_ORDER	869
7.108MATHS::CROSS_PRODUCT Interface Reference	870
7.108.1 Detailed Description	870
7.108.2 Member Function Documentation	870
7.108.2.1 CROSS_PRODUCT_DP	870
7.108.2.2 CROSS_PRODUCT_INTG	870
7.108.2.3 CROSS_PRODUCT_SP	870
7.109MATHS::D_CROSS_PRODUCT Interface Reference	871
7.109.1 Detailed Description	871
7.109.2 Member Function Documentation	871
7.109.2.1 D_CROSS_PRODUCT_DP	871
7.109.2.2 D_CROSS_PRODUCT_INTG	871
7.109.2.3 D_CROSS_PRODUCT_SP	871
7.110MATHS::DETERMINANT Interface Reference	872
7.110.1 Detailed Description	872
7.110.2 Member Function Documentation	872
7.110.2.1 DETERMINANT_FULL_DP	872
7.110.2.2 DETERMINANT_FULL_INTG	872
7.110.2.3 DETERMINANT_FULL_SP	872
7.111MATHS::EDP Interface Reference	873
7.111.1 Detailed Description	873
7.111.2 Member Function Documentation	873
7.111.2.1 EDP_DP	873
7.111.2.2 EDP_SP	873
7.112MATHS::EIGENVALUE Interface Reference	874
7.112.1 Detailed Description	874
7.112.2 Member Function Documentation	874
7.112.2.1 EIGENVALUE_FULL_DP	874
7.112.2.2 EIGENVALUE_FULL_SP	874
7.113MATHS::EIGENVECTOR Interface Reference	875

7.113.1 Detailed Description	875
7.113.2 Member Function Documentation	875
7.113.2.1 EIGENVECTOR_FULL_DP	875
7.113.2.2 EIGENVECTOR_FULL_SP	875
7.114MATHS::I0 Interface Reference	876
7.114.1 Detailed Description	876
7.114.2 Member Function Documentation	876
7.114.2.1 I0_DP	876
7.114.2.2 I0_SP	876
7.115MATHS::I1 Interface Reference	877
7.115.1 Detailed Description	877
7.115.2 Member Function Documentation	877
7.115.2.1 I1_DP	877
7.115.2.2 I1_SP	877
7.116MATHS::INVERT Interface Reference	878
7.116.1 Detailed Description	878
7.116.2 Member Function Documentation	878
7.116.2.1 INVERT_FULL_DP	878
7.116.2.2 INVERT_FULL_SP	878
7.117MATHS::K0 Interface Reference	879
7.117.1 Detailed Description	879
7.117.2 Member Function Documentation	879
7.117.2.1 K0_DP	879
7.117.2.2 K0_SP	879
7.118MATHS::K1 Interface Reference	880
7.118.1 Detailed Description	880
7.118.2 Member Function Documentation	880
7.118.2.1 K1_DP	880
7.118.2.2 K1_SP	880
7.119MATHS::L2NORM Interface Reference	881
7.119.1 Detailed Description	881
7.119.2 Member Function Documentation	881
7.119.2.1 L2NORM_DP	881
7.119.2.2 L2NORM_SP	881
7.120MATHS::NORMALISE Interface Reference	882
7.120.1 Detailed Description	882

7.120.2 Member Function Documentation	882
7.120.2.1 NORMALISE_DP	882
7.120.2.2 NORMALISE_SP	882
7.121 MATHS::SOLVE_SMALL_LINEAR_SYSTEM Interface Reference	883
7.121.1 Detailed Description	883
7.121.2 Member Function Documentation	883
7.121.2.1 SOLVE_SMALL_LINEAR_SYSTEM_DP	883
7.121.2.2 SOLVE_SMALL_LINEAR_SYSTEM_SP	883
7.122 MATRIX_VECTOR::MATRIX_ALL_VALUES_SET Interface Reference	884
7.122.1 Detailed Description	884
7.122.2 Member Function Documentation	884
7.122.2.1 MATRIX_ALL_VALUES_SET_DP	884
7.122.2.2 MATRIX_ALL_VALUES_SET_INTG	884
7.122.2.3 MATRIX_ALL_VALUES_SET_L	885
7.122.2.4 MATRIX_ALL_VALUES_SET_SP	885
7.123 MATRIX_VECTOR::MATRIX_DATA_GET Interface Reference	886
7.123.1 Detailed Description	886
7.123.2 Member Function Documentation	886
7.123.2.1 MATRIX_DATA_GET_DP	886
7.123.2.2 MATRIX_DATA_GET_INTG	886
7.123.2.3 MATRIX_DATA_GET_L	887
7.123.2.4 MATRIX_DATA_GET_SP	887
7.124 MATRIX_VECTOR::MATRIX_VALUES_ADD Interface Reference	888
7.124.1 Detailed Description	888
7.124.2 Member Function Documentation	888
7.124.2.1 MATRIX_VALUES_ADD_DP	888
7.124.2.2 MATRIX_VALUES_ADD_DP1	889
7.124.2.3 MATRIX_VALUES_ADD_DP2	889
7.124.2.4 MATRIX_VALUES_ADD_INTG	889
7.124.2.5 MATRIX_VALUES_ADD_INTG1	890
7.124.2.6 MATRIX_VALUES_ADD_INTG2	890
7.124.2.7 MATRIX_VALUES_ADD_L	891
7.124.2.8 MATRIX_VALUES_ADD_L1	891
7.124.2.9 MATRIX_VALUES_ADD_L2	891
7.124.2.10 MATRIX_VALUES_ADD_SP	892
7.124.2.11 MATRIX_VALUES_ADD_SP1	892

7.124.2.12MATRIX_VALUES_ADD_SP2	892
7.125MATRIX_VECTOR::MATRIX_VALUES_GET Interface Reference	893
7.125.1 Detailed Description	893
7.125.2 Member Function Documentation	893
7.125.2.1 MATRIX_VALUES_GET_DP	893
7.125.2.2 MATRIX_VALUES_GET_DP1	894
7.125.2.3 MATRIX_VALUES_GET_DP2	894
7.125.2.4 MATRIX_VALUES_GET_INTG	894
7.125.2.5 MATRIX_VALUES_GET_INTG1	895
7.125.2.6 MATRIX_VALUES_GET_INTG2	895
7.125.2.7 MATRIX_VALUES_GET_L	896
7.125.2.8 MATRIX_VALUES_GET_L1	896
7.125.2.9 MATRIX_VALUES_GET_L2	896
7.125.2.10MATRIX_VALUES_GET_SP	897
7.125.2.11MATRIX_VALUES_GET_SP1	897
7.125.2.12MATRIX_VALUES_GET_SP2	897
7.126MATRIX_VECTOR::MATRIX_VALUES_SET Interface Reference	898
7.126.1 Detailed Description	898
7.126.2 Member Function Documentation	898
7.126.2.1 MATRIX_VALUES_SET_DP	898
7.126.2.2 MATRIX_VALUES_SET_DP1	899
7.126.2.3 MATRIX_VALUES_SET_DP2	899
7.126.2.4 MATRIX_VALUES_SET_INTG	899
7.126.2.5 MATRIX_VALUES_SET_INTG1	900
7.126.2.6 MATRIX_VALUES_SET_INTG2	900
7.126.2.7 MATRIX_VALUES_SET_L	900
7.126.2.8 MATRIX_VALUES_SET_L1	901
7.126.2.9 MATRIX_VALUES_SET_L2	901
7.126.2.10MATRIX_VALUES_SET_SP	902
7.126.2.11MATRIX_VALUES_SET_SP1	902
7.126.2.12MATRIX_VALUES_SET_SP2	902
7.127MATRIX_VECTOR::VECTOR_ALL_VALUES_SET Interface Reference	903
7.127.1 Detailed Description	903
7.127.2 Member Function Documentation	903
7.127.2.1 VECTOR_ALL_VALUES_SET_DP	903
7.127.2.2 VECTOR_ALL_VALUES_SET_INTG	903

7.127.2.3 VECTOR_ALL_VALUES_SET_L	904
7.127.2.4 VECTOR_ALL_VALUES_SET_SP	904
7.128 MATRIX_VECTOR::VECTOR_DATA_GET Interface Reference	905
7.128.1 Detailed Description	905
7.128.2 Member Function Documentation	905
7.128.2.1 VECTOR_DATA_GET_DP	905
7.128.2.2 VECTOR_DATA_GET_INTG	905
7.128.2.3 VECTOR_DATA_GET_L	906
7.128.2.4 VECTOR_DATA_GET_SP	906
7.129 MATRIX_VECTOR::VECTOR_VALUES_GET Interface Reference	907
7.129.1 Detailed Description	907
7.129.2 Member Function Documentation	907
7.129.2.1 VECTOR_VALUES_GET_DP	907
7.129.2.2 VECTOR_VALUES_GET_DP1	907
7.129.2.3 VECTOR_VALUES_GET_INTG	908
7.129.2.4 VECTOR_VALUES_GET_INTG1	908
7.129.2.5 VECTOR_VALUES_GET_L	908
7.129.2.6 VECTOR_VALUES_GET_L1	909
7.129.2.7 VECTOR_VALUES_GET_SP	909
7.129.2.8 VECTOR_VALUES_GET_SP1	909
7.130 MATRIX_VECTOR::VECTOR_VALUES_SET Interface Reference	910
7.130.1 Detailed Description	910
7.130.2 Member Function Documentation	910
7.130.2.1 VECTOR_VALUES_SET_DP	910
7.130.2.2 VECTOR_VALUES_SET_DP1	910
7.130.2.3 VECTOR_VALUES_SET_INTG	911
7.130.2.4 VECTOR_VALUES_SET_INTG1	911
7.130.2.5 VECTOR_VALUES_SET_L	911
7.130.2.6 VECTOR_VALUES_SET_L1	912
7.130.2.7 VECTOR_VALUES_SET_SP	912
7.130.2.8 VECTOR_VALUES_SET_SP1	912
7.131 MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET Interface Reference	913
7.131.1 Detailed Description	913
7.131.2 Member Function Documentation	913
7.131.2.1 MESH_NUMBER_OF_COMPONENTS_SET_NUMBER	913
7.131.2.2 MESH_NUMBER_OF_COMPONENTS_SET_PTR	913

7.132MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET Interface Reference	914
7.132.1 Detailed Description	914
7.132.2 Member Function Documentation	914
7.132.2.1 MESH_NUMBER_OF_ELEMENTS_SET_NUMBER	914
7.132.2.2 MESH_NUMBER_OF_ELEMENTS_SET_PTR	914
7.133PROBLEM_ROUTINES::PROBLEM_DESTROY Interface Reference	915
7.133.1 Detailed Description	915
7.133.2 Member Function Documentation	915
7.133.2.1 PROBLEM_DESTROY_NUMBER	915
7.133.2.2 PROBLEM_DESTROY_PTR	915
7.134PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET Interface Reference	916
7.134.1 Detailed Description	916
7.134.2 Member Function Documentation	916
7.134.2.1 PROBLEM_SPECIFICATION_SET_NUMBER	916
7.134.2.2 PROBLEM_SPECIFICATION_SET_PTR	916
7.135REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET Interface Reference . .	918
7.135.1 Detailed Description	918
7.135.2 Member Function Documentation	918
7.135.2.1 REGION_COORDINATE_SYSTEM_SET_NUMBER	918
7.135.2.2 REGION_COORDINATE_SYSTEM_SET_PTR	918
7.136REGION_ROUTINES::REGION_LABEL_SET Interface Reference	919
7.136.1 Detailed Description	919
7.136.2 Member Function Documentation	919
7.136.2.1 REGION_LABEL_SET_NUMBER	919
7.136.2.2 REGION_LABEL_SET_PTR	919
7.137SORTING::BUBBLE_SORT Interface Reference	920
7.137.1 Detailed Description	920
7.137.2 Member Function Documentation	920
7.137.2.1 BUBBLE_SORT_DP	920
7.137.2.2 BUBBLE_SORT_INTG	920
7.137.2.3 BUBBLE_SORT_SP	920
7.138SORTING::HEAP_SORT Interface Reference	921
7.138.1 Detailed Description	921
7.138.2 Member Function Documentation	921
7.138.2.1 HEAP_SORT_DP	921
7.138.2.2 HEAP_SORT_INTG	921

7.138.2.3 HEAP_SORT_SP	921
7.139 SORTING::SHELL_SORT Interface Reference	922
7.139.1 Detailed Description	922
7.139.2 Member Function Documentation	922
7.139.2.1 SHELL_SORT_DP	922
7.139.2.2 SHELL_SORT_INTG	922
7.139.2.3 SHELL_SORT_SP	922
7.140 STRINGS::CHARACTER_TO_LOWERCASE Interface Reference	923
7.140.1 Detailed Description	923
7.140.2 Member Function Documentation	923
7.140.2.1 CHARACTER_TO_LOWERCASE_C	923
7.140.2.2 CHARACTER_TO_LOWERCASE_VS	923
7.141 STRINGS::CHARACTER_TO_UPPERCASE Interface Reference	924
7.141.1 Detailed Description	924
7.141.2 Member Function Documentation	924
7.141.2.1 CHARACTER_TO_UPPERCASE_C	924
7.141.2.2 CHARACTER_TO_UPPERCASE_VS	924
7.142 STRINGS::IS_ABBREVIATION Interface Reference	925
7.142.1 Detailed Description	925
7.142.2 Member Function Documentation	925
7.142.2.1 IS_ABBREVIATION_C_C	925
7.142.2.2 IS_ABBREVIATION_C_VS	925
7.142.2.3 IS_ABBREVIATION_VS_C	925
7.142.2.4 IS_ABBREVIATION_VS_VS	926
7.143 STRINGS::LIST_TO_CHARACTER Interface Reference	927
7.143.1 Detailed Description	927
7.143.2 Member Function Documentation	927
7.143.2.1 LIST_TO_CHARACTER_C	927
7.143.2.2 LIST_TO_CHARACTER_DP	928
7.143.2.3 LIST_TO_CHARACTER_INTG	928
7.143.2.4 LIST_TO_CHARACTER_L	928
7.143.2.5 LIST_TO_CHARACTER_LINTG	929
7.143.2.6 LIST_TO_CHARACTER_SP	929
7.144 STRINGS::NUMBER_TO_CHARACTER Interface Reference	930
7.144.1 Detailed Description	930
7.144.2 Member Function Documentation	930

7.144.2.1 NUMBER_TO_CHARACTER_DP	930
7.144.2.2 NUMBER_TO_CHARACTER_INTG	930
7.144.2.3 NUMBER_TO_CHARACTER_LINTG	931
7.144.2.4 NUMBER_TO_CHARACTER_SP	931
7.145 STRINGS::NUMBER_TO_VSTRING Interface Reference	932
7.145.1 Detailed Description	932
7.145.2 Member Function Documentation	932
7.145.2.1 NUMBER_TO_VSTRING_DP	932
7.145.2.2 NUMBER_TO_VSTRING_INTG	932
7.145.2.3 NUMBER_TO_VSTRING_LINTG	933
7.145.2.4 NUMBER_TO_VSTRING_SP	933
7.146 STRINGS::STRING_TO_DOUBLE Interface Reference	934
7.146.1 Detailed Description	934
7.146.2 Member Function Documentation	934
7.146.2.1 STRING_TO_DOUBLE_C	934
7.146.2.2 STRING_TO_DOUBLE_VS	934
7.147 STRINGS::STRING_TO_INTEGER Interface Reference	935
7.147.1 Detailed Description	935
7.147.2 Member Function Documentation	935
7.147.2.1 STRING_TO_INTEGER_C	935
7.147.2.2 STRING_TO_INTEGER_VS	935
7.148 STRINGS::STRING_TO_LOGICAL Interface Reference	936
7.148.1 Detailed Description	936
7.148.2 Member Function Documentation	936
7.148.2.1 STRING_TO_LOGICAL_C	936
7.148.2.2 STRING_TO_LOGICAL_VS	936
7.149 STRINGS::STRING_TO_LONG_INTEGER Interface Reference	937
7.149.1 Detailed Description	937
7.149.2 Member Function Documentation	937
7.149.2.1 STRING_TO_LONG_INTEGER_C	937
7.149.2.2 STRING_TO_LONG_INTEGER_VS	937
7.150 STRINGS::STRING_TO_SINGLE Interface Reference	938
7.150.1 Detailed Description	938
7.150.2 Member Function Documentation	938
7.150.2.1 STRING_TO_SINGLE_C	938
7.150.2.2 STRING_TO_SINGLE_VS	938

7.151STRINGS::VSTRING_TO_LOWER CASE Interface Reference	939
7.151.1 Detailed Description	939
7.151.2 Member Function Documentation	939
7.151.2.1 VSTRING_TO_LOWER CASE_C	939
7.151.2.2 VSTRING_TO_LOWER CASE_VS	939
7.152STRINGS::VSTRING_TO_UPPER CASE Interface Reference	940
7.152.1 Detailed Description	940
7.152.2 Member Function Documentation	940
7.152.2.1 VSTRING_TO_UPPER CASE_C	940
7.152.2.2 VSTRING_TO_UPPER CASE_VS	940
7.153TIMER::interface Interface Reference	941
7.153.1 Detailed Description	941
7.153.2 Member Function Documentation	941
7.153.2.1 CPUTIMER	941
7.154TREES::TREE_NODE_TYPE Struct Reference	942
7.154.1 Detailed Description	942
7.154.2 Member Data Documentation	942
7.154.2.1 COLOUR	942
7.154.2.2 KEY	942
7.154.2.3 LEFT	942
7.154.2.4 PARENT	943
7.154.2.5 RIGHT	943
7.154.2.6 VALUE	943
7.155TREES::TREE_TYPE Struct Reference	944
7.155.1 Detailed Description	944
7.155.2 Member Data Documentation	944
7.155.2.1 INSERT_TYPE	944
7.155.2.2 NIL	944
7.155.2.3 NUMBER_IN_TREE	944
7.155.2.4 ROOT	944
7.155.2.5 TREE_FINISHED	945
7.156TYPES::BASIS_FUNCTIONS_TYPE Struct Reference	946
7.156.1 Detailed Description	946
7.156.2 Member Data Documentation	946
7.156.2.1 BASES	946
7.156.2.2 NUMBER_BASIS_FUNCTIONS	946

7.157TYPES::BASIS_PTR_TYPE Struct Reference	947
7.157.1 Detailed Description	947
7.157.2 Member Data Documentation	947
7.157.2.1 PTR	947
7.158TYPES::BASIS_TYPE Struct Reference	948
7.158.1 Detailed Description	951
7.158.2 Member Data Documentation	951
7.158.2.1 BASIS_FINISHED	951
7.158.2.2 COLLAPSED_XI	951
7.158.2.3 DEGENERATE	951
7.158.2.4 DERIVATIVE_NUMBERS_IN_LOCAL_LINE	952
7.158.2.5 DERIVATIVE_ORDER_INDEX	952
7.158.2.6 DERIVATIVE_ORDER_INDEX_INV	952
7.158.2.7 ELEMENT_PARAMETER_INDEX	952
7.158.2.8 FACE_BASES	952
7.158.2.9 FAMILY_NUMBER	953
7.158.2.10GLOBAL_NUMBER	953
7.158.2.11HERMITE	953
7.158.2.12INTERPOLATION_ORDER	953
7.158.2.13INTERPOLATION_TYPE	953
7.158.2.14INTERPOLATION_XI	953
7.158.2.15LINE_BASES	954
7.158.2.16LOCAL_LINE_XI_DIRECTION	954
7.158.2.17MAXIMUM_NUMBER_OF_DERIVATIVES	954
7.158.2.18NODE_AT_COLLAPSE	954
7.158.2.19NODE_NUMBERS_IN_LOCAL_LINE	954
7.158.2.20NODE_POSITION_INDEX	954
7.158.2.21NODE_POSITION_INDEX_INV	955
7.158.2.22NUMBER_OF_COLLAPSED_XI	955
7.158.2.23NUMBER_OF_DERIVATIVES	955
7.158.2.24NUMBER_OF_ELEMENT_PARAMETERS	955
7.158.2.25NUMBER_OF_LOCAL_LINES	955
7.158.2.26NUMBER_OF_NODES	955
7.158.2.27NUMBER_OF_NODES_IN_LOCAL_LINE	955
7.158.2.28NUMBER_OF_NODES_XI	956
7.158.2.29NUMBER_OF_PARTIAL_DERIVATIVES	956

7.158.2.30NUMBER_OF_SUB_BASES	956
7.158.2.31NUMBER_OF_XI	956
7.158.2.32NUMBER_OF_XI_COORDINATES	956
7.158.2.33PARENT_BASIS	956
7.158.2.34PARTIAL_DERIVATIVE_INDEX	956
7.158.2.35QUADRATURE	957
7.158.2.36SUB_BASES	957
7.158.2.37TYPE	957
7.158.2.38USER_NUMBER	957
7.159TYPES::COORDINATE_SYSTEM_TYPE Struct Reference	958
7.159.1 Detailed Description	958
7.159.2 Member Data Documentation	959
7.159.2.1 COORDINATE_SYSTEM_FINISHED	959
7.159.2.2 FOCUS	959
7.159.2.3 NUMBER_OF_DIMENSIONS	959
7.159.2.4 ORIENTATION	959
7.159.2.5 ORIGIN	959
7.159.2.6 RADIAL_INTERPOLATION_TYPE	959
7.159.2.7 TYPE	959
7.159.2.8 USER_NUMBER	960
7.160TYPES::DECOMPOSITION_ELEMENT_TYPE Struct Reference	961
7.160.1 Detailed Description	961
7.160.2 Member Data Documentation	961
7.160.2.1 ADJACENT_ELEMENTS	961
7.160.2.2 ELEMENT_LINES	962
7.160.2.3 GLOBAL_NUMBER	962
7.160.2.4 LOCAL_NUMBER	962
7.160.2.5 NUMBER_OF_ADJACENT_ELEMENTS	962
7.160.2.6 USER_NUMBER	962
7.161TYPES::DECOMPOSITION_ELEMENTS_TYPE Struct Reference	963
7.161.1 Detailed Description	963
7.161.2 Member Data Documentation	963
7.161.2.1 DECOMPOSITION	963
7.161.2.2 ELEMENTS	963
7.161.2.3 TOTAL_NUMBER_OF_ELEMENTS	964
7.162TYPES::DECOMPOSITION_LINE_TYPE Struct Reference	965

7.162.1 Detailed Description	965
7.162.2 Member Data Documentation	965
7.162.2.1 ADJACENT_LINES	965
7.162.2.2 ELEMENT_LINES	965
7.162.2.3 NUMBER	966
7.162.2.4 NUMBER_OF_SURROUNDING_ELEMENTS	966
7.162.2.5 SURROUNDING_ELEMENTS	966
7.162.2.6 XI_DIRECTION	966
7.163TYPES::DECOMPOSITION_LINES_TYPE Struct Reference	967
7.163.1 Detailed Description	967
7.163.2 Member Data Documentation	967
7.163.2.1 DECOMPOSITION	967
7.163.2.2 LINES	967
7.163.2.3 NUMBER_OF_LINES	967
7.164TYPES::DECOMPOSITION_PTR_TYPE Struct Reference	968
7.164.1 Detailed Description	968
7.164.2 Member Data Documentation	968
7.164.2.1 PTR	968
7.165TYPES::DECOMPOSITION_TOPOLOGY_TYPE Struct Reference	969
7.165.1 Detailed Description	969
7.165.2 Member Data Documentation	969
7.165.2.1 DECOMPOSITION	969
7.165.2.2 ELEMENTS	969
7.165.2.3 LINES	969
7.166TYPES::DECOMPOSITION_TYPE Struct Reference	970
7.166.1 Detailed Description	971
7.166.2 Member Data Documentation	971
7.166.2.1 DECOMPOSITION_FINISHED	971
7.166.2.2 DECOMPOSITION_TYPE	971
7.166.2.3 DECOMPOSITIONS	971
7.166.2.4 DOMAIN	971
7.166.2.5 ELEMENT_DOMAIN	971
7.166.2.6 GLOBAL_NUMBER	972
7.166.2.7 MESH	972
7.166.2.8 MESH_COMPONENT_NUMBER	972
7.166.2.9 NUMBER_OF_DOMAINS	972

7.166.2.10	NUMBER_OF_EDGES_CUT	972
7.166.2.11	TOPOLOGY	972
7.166.2.12	USER_NUMBER	972
7.167	TYPES::DECOMPOSITIONS_TYPE Struct Reference	973
7.167.1	Detailed Description	973
7.167.2	Member Data Documentation	973
7.167.2.1	DECOMPOSITIONS	973
7.167.2.2	MESH	973
7.167.2.3	NUMBER_OF_DECOMPOSITIONS	973
7.168	TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE Struct Reference	974
7.168.1	Detailed Description	974
7.168.2	Member Data Documentation	974
7.168.2.1	BASE_TAG_NUMBER	974
7.168.2.2	DISTRIBUTED_MATRIX	974
7.168.2.3	MATRIX	974
7.169	TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE Struct Reference	975
7.169.1	Detailed Description	976
7.169.2	Member Data Documentation	976
7.169.2.1	COLUMN_INDICES	976
7.169.2.2	DATA_DP	976
7.169.2.3	DATA_SIZE	976
7.169.2.4	DIAGONAL_NUMBER_NON_ZEROS	976
7.169.2.5	DISTRIBUTED_MATRIX	976
7.169.2.6	GLOBAL_M	977
7.169.2.7	GLOBAL_N	977
7.169.2.8	GLOBAL_ROW_NUMBERS	977
7.169.2.9	ISLTGMAPPING	977
7.169.2.10	M	977
7.169.2.11	IMATRIX	977
7.169.2.12	MAXIMUM_COLUMN_INDICES_PER_ROW	977
7.169.2.13	N	977
7.169.2.14	NUMBER_NON_ZEROS	978
7.169.2.15	OFFDIAGONAL_NUMBER_NON_ZEROS	978
7.169.2.16	ROW_INDICES	978
7.169.2.17	STORAGE_TYPE	978
7.170	TYPES::DISTRIBUTED_MATRIX_TYPE Struct Reference	979

7.170.1 Detailed Description	979
7.170.2 Member Data Documentation	979
7.170.2.1 CMISS	979
7.170.2.2 COLUMN_DOMAIN_MAPPING	980
7.170.2.3 DATA_TYPE	980
7.170.2.4 GHOSTING_TYPE	980
7.170.2.5 LIBRARY_TYPE	980
7.170.2.6 MATRIX_FINISHED	980
7.170.2.7 PETSC	980
7.170.2.8 ROW_DOMAIN_MAPPING	981
7.171TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE Struct Reference	982
7.171.1 Detailed Description	982
7.171.2 Member Data Documentation	983
7.171.2.1 BASE_TAG_NUMBER	983
7.171.2.2 DATA_DP	983
7.171.2.3 DATA_INTG	983
7.171.2.4 DATA_L	983
7.171.2.5 DATA_SIZE	983
7.171.2.6 DATA_SP	983
7.171.2.7 DISTRIBUTED_VECTOR	983
7.171.2.8 N	984
7.171.2.9 TRANSFERS	984
7.172TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE Struct Reference	985
7.172.1 Detailed Description	985
7.172.2 Member Data Documentation	985
7.172.2.1 DATA_SIZE	985
7.172.2.2 DISTRIBUTED_VECTOR	985
7.172.2.3 GLOBAL_N	986
7.172.2.4 GLOBAL_NUMBERS	986
7.172.2.5 ISLTGMAPPING	986
7.172.2.6 N	986
7.172.2.7 VECTOR	986
7.173TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE Struct Reference	987
7.173.1 Detailed Description	988
7.173.2 Member Data Documentation	988
7.173.2.1 CMISS_VECTOR	988

7.173.2.2 DATA_TYPE	988
7.173.2.3 MPI_RECEIVE_REQUEST	988
7.173.2.4 MPI_SEND_REQUEST	988
7.173.2.5 RECEIVE_BUFFER_DP	989
7.173.2.6 RECEIVE_BUFFER_INTG	989
7.173.2.7 RECEIVE_BUFFER_L	989
7.173.2.8 RECEIVE_BUFFER_SIZE	989
7.173.2.9 RECEIVE_BUFFER_SP	989
7.173.2.10 RECEIVE_TAG_NUMBER	989
7.173.2.11 SEND_BUFFER_DP	990
7.173.2.12 SEND_BUFFER_INTG	990
7.173.2.13 SEND_BUFFER_L	990
7.173.2.14 SEND_BUFFER_SIZE	990
7.173.2.15 SEND_BUFFER_SP	990
7.173.2.16 SEND_TAG_NUMBER	990
7.174 TYPES::DISTRIBUTED_VECTOR_TYPE Struct Reference	991
7.174.1 Detailed Description	991
7.174.2 Member Data Documentation	991
7.174.2.1 CMISS	991
7.174.2.2 DATA_TYPE	991
7.174.2.3 DOMAIN_MAPPING	992
7.174.2.4 GHOSTING_TYPE	992
7.174.2.5 LIBRARY_TYPE	992
7.174.2.6 PETSC	992
7.174.2.7 VECTOR_FINISHED	992
7.175 TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE Struct Reference	993
7.175.1 Detailed Description	993
7.175.2 Member Data Documentation	993
7.175.2.1 DOMAIN_NUMBER	993
7.175.2.2 LOCAL_GHOST_RECEIVE_INDICES	993
7.175.2.3 LOCAL_GHOST_SEND_INDICES	994
7.175.2.4 NUMBER_OF_RECEIVE_GHOSTS	994
7.175.2.5 NUMBER_OF_SEND_GHOSTS	994
7.176 TYPES::DOMAIN_DOFS_TYPE Struct Reference	995
7.176.1 Detailed Description	995
7.176.2 Member Data Documentation	995

7.176.2.1 DOF_INDEX	995
7.176.2.2 DOMAIN	995
7.176.2.3 NUMBER_OF_DOFS	995
7.176.2.4 TOTAL_NUMBER_OF_DOFS	996
7.177 TYPES::DOMAIN_ELEMENT_TYPE Struct Reference	997
7.177.1 Detailed Description	997
7.177.2 Member Data Documentation	997
7.177.2.1 BASIS	997
7.177.2.2 ELEMENT_DERIVATIVES	997
7.177.2.3 ELEMENT_NODES	997
7.177.2.4 NUMBER	998
7.178 TYPES::DOMAIN_ELEMENTS_TYPE Struct Reference	999
7.178.1 Detailed Description	999
7.178.2 Member Data Documentation	999
7.178.2.1 DOMAIN	999
7.178.2.2 ELEMENTS	999
7.178.2.3 MAXIMUM_NUMBER_OF_ELEMENT_PARAMETERS	1000
7.178.2.4 NUMBER_OF_ELEMENTS	1000
7.178.2.5 TOTAL_NUMBER_OF_ELEMENTS	1000
7.179 TYPES::DOMAIN_FACE_PTR_TYPE Struct Reference	1001
7.179.1 Detailed Description	1001
7.179.2 Member Data Documentation	1001
7.179.2.1 PTR	1001
7.180 TYPES::DOMAIN_FACE_TYPE Struct Reference	1002
7.180.1 Detailed Description	1002
7.180.2 Member Data Documentation	1002
7.180.2.1 BASIS	1002
7.180.2.2 DERIVATIVES_IN_FACE	1002
7.180.2.3 NODES_IN_FACE	1003
7.180.2.4 NUMBER	1003
7.180.2.5 XI_DIRECTION1	1003
7.180.2.6 XI_DIRECTION2	1003
7.181 TYPES::DOMAIN_FACES_TYPE Struct Reference	1004
7.181.1 Detailed Description	1004
7.181.2 Member Data Documentation	1004
7.181.2.1 DOMAIN	1004

7.181.2.2 FACES	1004
7.181.2.3 NUMBER_OF_FACES	1004
7.182TYPES::DOMAIN_GLOBAL_MAPPING_TYPE Struct Reference	1005
7.182.1 Detailed Description	1005
7.182.2 Member Data Documentation	1005
7.182.2.1 DOMAIN_NUMBER	1005
7.182.2.2 LOCAL_NUMBER	1005
7.182.2.3 LOCAL_TYPE	1006
7.182.2.4 NUMBER_OF_DOMAINS	1006
7.183TYPES::DOMAIN_LINE_PTR_TYPE Struct Reference	1007
7.183.1 Detailed Description	1007
7.183.2 Member Data Documentation	1007
7.183.2.1 PTR	1007
7.184TYPES::DOMAIN_LINE_TYPE Struct Reference	1008
7.184.1 Detailed Description	1008
7.184.2 Member Data Documentation	1008
7.184.2.1 BASIS	1008
7.184.2.2 DERIVATIVES_IN_LINE	1008
7.184.2.3 NODES_IN_LINE	1008
7.184.2.4 NUMBER	1009
7.185TYPES::DOMAIN_LINES_TYPE Struct Reference	1010
7.185.1 Detailed Description	1010
7.185.2 Member Data Documentation	1010
7.185.2.1 DOMAIN	1010
7.185.2.2 LINES	1010
7.185.2.3 NUMBER_OF_LINES	1010
7.186TYPES::DOMAIN_MAPPING_TYPE Struct Reference	1011
7.186.1 Detailed Description	1012
7.186.2 Member Data Documentation	1012
7.186.2.1 ADJACENT_DOMAINS	1012
7.186.2.2 ADJACENT_DOMAINS_LIST	1012
7.186.2.3 ADJACENT_DOMAINS_PTR	1012
7.186.2.4 BOUNDARY_LIST	1013
7.186.2.5 GHOST_LIST	1013
7.186.2.6 GLOBAL_TO_LOCAL_MAP	1013
7.186.2.7 INTERNAL_LIST	1013

7.186.2.8 LOCAL_TO_GLOBAL_MAP	1013
7.186.2.9 NUMBER_OF_ADJACENT_DOMAINS	1013
7.186.2.10 NUMBER_OF_BOUNDARY	1014
7.186.2.11 NUMBER_OF_DOMAIN_LOCAL	1014
7.186.2.12 NUMBER_OF_DOMAINS	1014
7.186.2.13 NUMBER_OF_GHOST	1014
7.186.2.14 NUMBER_OF_GLOBAL	1014
7.186.2.15 NUMBER_OF_INTERNAL	1014
7.186.2.16 NUMBER_OF_LOCAL	1014
7.186.2.17 TOTAL_NUMBER_OF_LOCAL	1014
7.187 TYPES::DOMAIN_MAPPINGS_TYPE Struct Reference	1015
7.187.1 Detailed Description	1015
7.187.2 Member Data Documentation	1015
7.187.2.1 DOFS	1015
7.187.2.2 DOMAIN	1015
7.187.2.3 ELEMENTS	1015
7.187.2.4 NODES	1016
7.188 TYPES::DOMAIN_NODE_TYPE Struct Reference	1017
7.188.1 Detailed Description	1017
7.188.2 Member Data Documentation	1018
7.188.2.1 DOF_INDEX	1018
7.188.2.2 GLOBAL_NUMBER	1018
7.188.2.3 LOCAL_NUMBER	1018
7.188.2.4 NODE_LINES	1018
7.188.2.5 NUMBER_OF_DERIVATIVES	1018
7.188.2.6 NUMBER_OF_NODE_LINES	1018
7.188.2.7 NUMBER_OF_SURROUNDING_ELEMENTS	1018
7.188.2.8 PARTIAL_DERIVATIVE_INDEX	1018
7.188.2.9 SURROUNDING_ELEMENTS	1019
7.188.2.10 USER_NUMBER	1019
7.189 TYPES::DOMAIN_NODES_TYPE Struct Reference	1020
7.189.1 Detailed Description	1020
7.189.2 Member Data Documentation	1020
7.189.2.1 DOMAIN	1020
7.189.2.2 MAXIMUM_NUMBER_OF_DERIVATIVES	1020
7.189.2.3 NODES	1020

7.189.2.4 NUMBER_OF_NODES	1021
7.189.2.5 TOTAL_NUMBER_OF_NODES	1021
7.190 TYPES::DOMAIN_PTR_TYPE Struct Reference	1022
7.190.1 Detailed Description	1022
7.190.2 Member Data Documentation	1022
7.190.2.1 PTR	1022
7.191 TYPES::DOMAIN_TOPOLOGY_TYPE Struct Reference	1023
7.191.1 Detailed Description	1023
7.191.2 Member Data Documentation	1023
7.191.2.1 DOFS	1023
7.191.2.2 DOMAIN	1023
7.191.2.3 ELEMENTS	1024
7.191.2.4 FACES	1024
7.191.2.5 LINES	1024
7.191.2.6 NODES	1024
7.192 TYPES::DOMAIN_TYPE Struct Reference	1025
7.192.1 Detailed Description	1025
7.192.2 Member Data Documentation	1025
7.192.2.1 DECOMPOSITION	1025
7.192.2.2 MAPPINGS	1026
7.192.2.3 MESH	1026
7.192.2.4 MESH_COMPONENT_NUMBER	1026
7.192.2.5 NODE_DOMAIN	1026
7.192.2.6 NUMBER_OF_DIMENSIONS	1026
7.192.2.7 REGION	1026
7.192.2.8 TOPOLOGY	1026
7.193 TYPES::EIGENPROBLEM_SOLVER_TYPE Struct Reference	1027
7.193.1 Detailed Description	1027
7.193.2 Member Data Documentation	1027
7.193.2.1 SOLVER	1027
7.193.2.2 SOLVER_LIBRARY	1027
7.194 TYPES::ELEMENT_MATRIX_TYPE Struct Reference	1028
7.194.1 Detailed Description	1028
7.194.2 Member Data Documentation	1028
7.194.2.1 COLUMN_DOFS	1028
7.194.2.2 EQUATIONS_MATRIX_NUMBER	1028

7.194.2.3 MATRIX	1028
7.194.2.4 MAX_NUMBER_OF_COLUMNS	1028
7.194.2.5 MAX_NUMBER_OF_ROWS	1028
7.194.2.6 NUMBER_OF_COLUMNS	1029
7.194.2.7 NUMBER_OF_ROWS	1029
7.194.2.8 ROW_DOFS	1029
7.195 TYPES::ELEMENT_VECTOR_TYPE Struct Reference	1030
7.195.1 Detailed Description	1030
7.195.2 Member Data Documentation	1030
7.195.2.1 MAX_NUMBER_OF_ROWS	1030
7.195.2.2 NUMBER_OF_ROWS	1030
7.195.2.3 ROW_DOFS	1030
7.195.2.4 VECTOR	1030
7.196 TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE Struct Reference	1031
7.196.1 Detailed Description	1031
7.196.2 Member Data Documentation	1031
7.196.2.1 COUPLING_COEFFICIENTS	1031
7.196.2.2 NUMBER_OF_SOLVER_COLS	1031
7.196.2.3 SOLVER_COLS	1031
7.197 TYPES::EQUATIONS_INTERPOLATION_TYPE Struct Reference	1032
7.197.1 Detailed Description	1033
7.197.2 Member Data Documentation	1033
7.197.2.1 DEPENDENT_FIELD	1033
7.197.2.2 DEPENDENT_INTERP_PARAMETERS	1033
7.197.2.3 DEPENDENT_INTERP_POINT	1033
7.197.2.4 EQUATIONS	1034
7.197.2.5 FIBRE_FIELD	1034
7.197.2.6 FIBRE_INTERP_PARAMETERS	1034
7.197.2.7 FIBRE_INTERP_POINT	1034
7.197.2.8 FIBRE_INTERP_POINT_METRICS	1034
7.197.2.9 GEOMETRIC_FIELD	1034
7.197.2.10 GEOMETRIC_INTERP_PARAMETERS	1034
7.197.2.11 GEOMETRIC_INTERP_POINT	1035
7.197.2.12 GEOMETRIC_INTERP_POINT_METRICS	1035
7.197.2.13 MATERIAL_FIELD	1035
7.197.2.14 MATERIAL_INTERP_PARAMETERS	1035

7.197.2.15MATERIAL_INTERP_POINT	1035
7.197.2.16SOURCE_FIELD	1035
7.197.2.17SOURCE_INTERP_PARAMETERS	1035
7.197.2.18SOURCE_INTERP_POINT	1036
7.198TYPES::EQUATIONS_LINEAR_DATA_TYPE Struct Reference	1037
7.198.1 Detailed Description	1037
7.198.2 Member Data Documentation	1037
7.198.2.1 EQUATIONS	1037
7.199TYPES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE Struct Reference	1038
7.199.1 Detailed Description	1038
7.199.2 Member Data Documentation	1038
7.199.2.1 MATRIX_COEFFICIENTS	1038
7.199.2.2 MATRIX_VARIABLE_TYPES	1038
7.200TYPES::EQUATIONS_MAPPING_TYPE Struct Reference	1039
7.200.1 Detailed Description	1040
7.200.2 Member Data Documentation	1040
7.200.2.1 CREATE_VALUES_CACHE	1040
7.200.2.2 EQUATIONS	1040
7.200.2.3 EQUATIONS_MAPPING_FINISHED	1040
7.200.2.4 EQUATIONS_MATRICES	1040
7.200.2.5 EQUATIONS_MATRIX_TO_VARIABLE_MAPS	1041
7.200.2.6 EQUATIONS_ROW_TO_VARIABLES_MAPS	1041
7.200.2.7 MATRIX_VARIABLE_TYPES	1041
7.200.2.8 NUMBER_OF_EQUATIONS_MATRICES	1041
7.200.2.9 NUMBER_OF_MATRIX_VARIABLES	1041
7.200.2.10NUMBER_OF_ROWS	1041
7.200.2.11RHIS_VARIABLE	1041
7.200.2.12RHS_VARIABLE_MAPPING	1042
7.200.2.13RHS_VARIABLE_TYPE	1042
7.200.2.14ROW_DOFS_MAPPING	1042
7.200.2.15SOURCE_MAPPINGS	1042
7.200.2.16TOTAL_NUMBER_OF_ROWS	1042
7.200.2.17VARIABLE_TO_EQUATIONS_MATRICES_MAPS	1042
7.201TYPES::EQUATIONS_MATRICES_TYPE Struct Reference	1043
7.201.1 Detailed Description	1043
7.201.2 Member Data Documentation	1044

7.201.2.1 ELEMENT_VECTOR	1044
7.201.2.2 EQUATIONS	1044
7.201.2.3 EQUATIONS_MAPPING	1044
7.201.2.4 EQUATIONS_MATRICES_FINISHED	1044
7.201.2.5 MATRICES	1044
7.201.2.6 NUMBER_OF_MATRICES	1044
7.201.2.7 NUMBER_OF_ROWS	1044
7.201.2.8 SOLUTION_MAPPING	1045
7.201.2.9 TOTAL_NUMBER_OF_ROWS	1045
7.201.2.10 UPDATE_VECTOR	1045
7.201.2.11 VECTOR	1045
7.202 TYPES::EQUATIONS_MATRIX_PTR_TYPE Struct Reference	1046
7.202.1 Detailed Description	1046
7.202.2 Member Data Documentation	1046
7.202.2.1 PTR	1046
7.203 TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE Struct Reference	1047
7.203.1 Detailed Description	1047
7.203.2 Member Data Documentation	1047
7.203.2.1 COLUMN_DOFS_MAPPING	1047
7.203.2.2 COLUMN_TO_DOF_MAP	1048
7.203.2.3 EQUATIONS_MATRIX	1048
7.203.2.4 MATRIX_COEFFICIENT	1048
7.203.2.5 MATRIX_NUMBER	1048
7.203.2.6 NUMBER_OF_COLUMNS	1048
7.203.2.7 VARIABLE	1048
7.203.2.8 VARIABLE_TYPE	1048
7.204 TYPES::EQUATIONS_MATRIX_TYPE Struct Reference	1049
7.204.1 Detailed Description	1049
7.204.2 Member Data Documentation	1049
7.204.2.1 ELEMENT_MATRIX	1049
7.204.2.2 EQUATIONS_MATRICES	1050
7.204.2.3 MATRIX	1050
7.204.2.4 MATRIX_NUMBER	1050
7.204.2.5 NUMBER_OF_COLUMNS	1050
7.204.2.6 STORAGE_TYPE	1050
7.204.2.7 STRUCTURE_TYPE	1050

7.204.2.8 UPDATE_MATRIX	1050
7.205TYPES::EQUATIONS_NONLINEAR_DATA_TYPE Struct Reference	1051
7.205.1 Detailed Description	1051
7.205.2 Member Data Documentation	1051
7.205.2.1 EQUATIONS	1051
7.206TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE Struct Reference . . .	1052
7.206.1 Detailed Description	1052
7.206.2 Member Data Documentation	1052
7.206.2.1 COUPLING_COEFFICIENTS	1052
7.206.2.2 NUMBER_OF_SOLVER_ROWS	1052
7.206.2.3 SOLVER_ROWS	1052
7.207TYPES::EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE Struct Reference	1053
7.207.1 Detailed Description	1053
7.207.2 Member Data Documentation	1053
7.207.2.1 ROW_TO_DOFS_MAP	1053
7.207.2.2 ROW_TO_RHS_DOF	1053
7.208TYPES::EQUATIONS_SET_ANALYTIC_TYPE Struct Reference	1054
7.208.1 Detailed Description	1054
7.208.2 Member Data Documentation	1054
7.208.2.1 ANALYTIC_FINISHED	1054
7.208.2.2 EQUATIONS_SET	1054
7.209TYPES::EQUATIONS_SET_DEPENDENT_TYPE Struct Reference	1055
7.209.1 Detailed Description	1055
7.209.2 Member Data Documentation	1055
7.209.2.1 DEPENDENT_FIELD	1055
7.209.2.2 DEPENDENT_FINISHED	1055
7.209.2.3 EQUATIONS_SET	1055
7.210TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE Struct Reference	1056
7.210.1 Detailed Description	1056
7.210.2 Member Data Documentation	1056
7.210.2.1 BOUNDARY_CONDITIONS	1056
7.210.2.2 EQUATIONS_SET	1056
7.210.2.3 FIXED_CONDITIONS_FINISHED	1056
7.210.2.4 GLOBAL_BOUNDARY_CONDITIONS	1057
7.211TYPES::EQUATIONS_SET_GEOMETRY_TYPE Struct Reference	1058
7.211.1 Detailed Description	1058

7.211.2 Member Data Documentation	1058
7.211.2.1 EQUATIONS_SET	1058
7.211.2.2 FIBRE_FIELD	1058
7.211.2.3 GEOMETRIC_FIELD	1058
7.212TYPES::EQUATIONS_SET_MATERIALS_TYPE Struct Reference	1059
7.212.1 Detailed Description	1059
7.212.2 Member Data Documentation	1059
7.212.2.1 EQUATIONS_SET	1059
7.212.2.2 MATERIAL_FIELD	1059
7.212.2.3 MATERIALS_FINISHED	1059
7.213TYPES::EQUATIONS_SET_PTR_TYPE Struct Reference	1060
7.213.1 Detailed Description	1060
7.213.2 Member Data Documentation	1060
7.213.2.1 PTR	1060
7.214TYPES::EQUATIONS_SET_SOURCE_TYPE Struct Reference	1061
7.214.1 Detailed Description	1061
7.214.2 Member Data Documentation	1061
7.214.2.1 EQUATIONS_SET	1061
7.214.2.2 SOURCE_FIELD	1061
7.214.2.3 SOURCE_FINISHED	1061
7.215TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE Struct Reference	1062
7.215.1 Detailed Description	1062
7.215.2 Member Data Documentation	1062
7.215.2.1 EQUATIONS	1062
7.215.2.2 EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS	1063
7.215.2.3 EQUATIONS_SET_INDEX	1063
7.215.2.4 EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM	1063
7.215.2.5 EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM	1063
7.215.2.6 SOLUTION_MAPPING	1063
7.216TYPES::EQUATIONS_SET_TYPE Struct Reference	1064
7.216.1 Detailed Description	1065
7.216.2 Member Data Documentation	1065
7.216.2.1 ANALYTIC	1065
7.216.2.2 CLASS	1065
7.216.2.3 DEPENDENT	1065
7.216.2.4 EQUATIONS	1065

7.216.2.5 EQUATIONS_SET_FINISHED	1065
7.216.2.6 EQUATIONS_SETS	1066
7.216.2.7 FIXED_CONDITIONS	1066
7.216.2.8 GEOMETRY	1066
7.216.2.9 GLOBAL_NUMBER	1066
7.216.2.10 LINEARITY	1066
7.216.2.11 MATERIALS	1066
7.216.2.12 REGION	1066
7.216.2.13 SOLUTION_METHOD	1067
7.216.2.14 SOURCE	1067
7.216.2.15 SUBTYPE	1067
7.216.2.16 TIME_TYPE	1067
7.216.2.17 TYPE	1067
7.216.2.18 USER_NUMBER	1067
7.217 TYPES::EQUATIONS_SETS_TYPE Struct Reference	1068
7.217.1 Detailed Description	1068
7.217.2 Member Data Documentation	1068
7.217.2.1 EQUATIONS_SETS	1068
7.217.2.2 NUMBER_OF_EQUATIONS_SETS	1068
7.217.2.3 REGION	1068
7.218 TYPES::EQUATIONS_TIME_DATA_TYPE Struct Reference	1069
7.218.1 Detailed Description	1069
7.218.2 Member Data Documentation	1069
7.218.2.1 EQUATIONS	1069
7.219 TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE Struct Reference	1070
7.219.1 Detailed Description	1070
7.219.2 Member Data Documentation	1070
7.219.2.1 PTR	1070
7.220 TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE Struct Reference	1071
7.220.1 Detailed Description	1071
7.220.2 Member Data Documentation	1071
7.220.2.1 EQUATIONS_COL_SOLVER_COLS_MAP	1071
7.220.2.2 EQUATIONS_MATRIX	1071
7.220.2.3 EQUATIONS_MATRIX_NUMBER	1072
7.220.2.4 SOLVER_MATRIX	1072
7.220.2.5 SOLVER_MATRIX_NUMBER	1072

7.221	TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE Struct Reference	1073
7.221.1	Detailed Description	1073
7.221.2	Member Data Documentation	1073
7.221.2.1	EQUATIONS_MATRIX_NUMBER	1073
7.221.2.2	EQUATIONS_TO_SOLVER_MATRIX_MAPS	1073
7.221.2.3	NUMBER_OF_SOLVER_MATRICES	1073
7.222	TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE Struct Reference	1075
7.222.1	Detailed Description	1075
7.222.2	Member Data Documentation	1075
7.222.2.1	EQUATIONS_TO_SOLVER_MATRIX_MAPS	1075
7.222.2.2	NUMBER_OF_EQUATIONS_MATRICES	1076
7.222.2.3	NUMBER_OF_VARIABLES	1076
7.222.2.4	SOLVER_MATRIX_NUMBER	1076
7.222.2.5	VARIABLE_TO_SOLVER_COL_MAPS	1076
7.222.2.6	VARIABLE_TYPES	1076
7.222.2.7	VARIABLES	1076
7.223	TYPES::EQUATIONS_TYPE Struct Reference	1077
7.223.1	Detailed Description	1077
7.223.2	Member Data Documentation	1077
7.223.2.1	EQUATIONS_FINISHED	1077
7.223.2.2	EQUATIONS_MAPPING	1078
7.223.2.3	EQUATIONS_MATRICES	1078
7.223.2.4	EQUATIONS_SET	1078
7.223.2.5	INTERPOLATION	1078
7.223.2.6	LINEAR_DATA	1078
7.223.2.7	NONLINEAR_DATA	1078
7.223.2.8	OUTPUT_TYPE	1078
7.223.2.9	SPARSITY_TYPE	1079
7.223.2.10	TIME_DATA	1079
7.224	TYPES::FIELD_CREATE_VALUES_CACHE_TYPE Struct Reference	1080
7.224.1	Detailed Description	1080
7.224.2	Member Data Documentation	1080
7.224.2.1	INTERPOLATION_TYPE	1080
7.224.2.2	MESH_COMPONENT_NUMBER	1080
7.224.2.3	NUMBER_OF_COMPONENTS	1081
7.224.2.4	VARIABLE_TYPES	1081

7.225TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE Struct Reference	1082
7.225.1 Detailed Description	1083
7.225.2 Member Data Documentation	1083
7.225.2.1 CONSTANT_DOF2PARAM_MAP	1083
7.225.2.2 DOF_TYPE	1083
7.225.2.3 ELEMENT_DOF2PARAM_MAP	1083
7.225.2.4 NODE_DOF2PARAM_MAP	1084
7.225.2.5 NUMBER_OF_CONSTANT_DOFS	1084
7.225.2.6 NUMBER_OF_DOFS	1084
7.225.2.7 NUMBER_OF_ELEMENT_DOFS	1084
7.225.2.8 NUMBER_OF_NODE_DOFS	1084
7.225.2.9 NUMBER_OF_POINT_DOFS	1084
7.225.2.10POINT_DOF2PARAM_MAP	1084
7.225.2.11VARIABLE_DOF	1085
7.226TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE Struct Reference	1086
7.226.1 Detailed Description	1086
7.226.2 Member Data Documentation	1086
7.226.2.1 AREAS	1086
7.226.2.2 FIELDS_USING	1087
7.226.2.3 LENGTHS	1087
7.226.2.4 NUMBER_OF AREAS	1087
7.226.2.5 NUMBER_OF_FIELDS_USING	1087
7.226.2.6 NUMBER_OF_LINES	1087
7.226.2.7 NUMBER_OF_VOLUMES	1087
7.226.2.8 VOLUMES	1087
7.227TYPES::FIELD_INTERPOLATED_POINT_METRICS_TYPE Struct Reference	1088
7.227.1 Detailed Description	1088
7.227.2 Member Data Documentation	1088
7.227.2.1 DX_DXI	1088
7.227.2.2 DXI_DX	1089
7.227.2.3 GL	1089
7.227.2.4 GU	1089
7.227.2.5 INTERPOLATED_POINT	1089
7.227.2.6 JACOBIAN	1089
7.227.2.7 JACOBIAN_TYPE	1089
7.227.2.8 NUMBER_OF_X_DIMENSIONS	1089

7.227.2.9 NUMBER_OF_XI_DIMENSIONS	1090
7.228TYPES::FIELD_INTERPOLATED_POINT_TYPE Struct Reference	1091
7.228.1 Detailed Description	1091
7.228.2 Member Data Documentation	1091
7.228.2.1 INTERPOLATION_PARAMETERS	1091
7.228.2.2 MAX_PARTIAL_DERIVATIVE_INDEX	1091
7.228.2.3 PARTIAL_DERIVATIVE_TYPE	1092
7.228.2.4 VALUES	1092
7.229TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE Struct Reference	1093
7.229.1 Detailed Description	1093
7.229.2 Member Data Documentation	1093
7.229.2.1 BASES	1093
7.229.2.2 FIELD	1093
7.229.2.3 FIELD_VARIABLE	1094
7.229.2.4 NUMBER_OF_PARAMETERS	1094
7.229.2.5 PARAMETERS	1094
7.230TYPES::FIELD_MAPPINGS_TYPE Struct Reference	1095
7.230.1 Detailed Description	1095
7.230.2 Member Data Documentation	1095
7.230.2.1 DOF_TO_PARAM_MAP	1095
7.230.2.2 DOMAIN_MAPPING	1095
7.231TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE Struct Reference	1096
7.231.1 Detailed Description	1096
7.231.2 Member Data Documentation	1097
7.231.2.1 CONSTANT_PARAM2DOF_MAP	1097
7.231.2.2 ELEMENT_PARAM2DOF_MAP	1097
7.231.2.3 MAX_NUMBER_OF_DERIVATIVES	1097
7.231.2.4 NODE_PARAM2DOF_MAP	1097
7.231.2.5 NUMBER_OF_CONSTANT_PARAMETERS	1097
7.231.2.6 NUMBER_OF_ELEMENT_PARAMETERS	1098
7.231.2.7 NUMBER_OF_NODE_PARAMETERS	1098
7.231.2.8 NUMBER_OF_POINT_PARAMETERS	1098
7.231.2.9 POINT_PARAM2DOF_MAP	1098
7.232TYPES::FIELD_PARAMETER_SET_PTR_TYPE Struct Reference	1099
7.232.1 Detailed Description	1099
7.232.2 Member Data Documentation	1099

7.232.2.1 PTR	1099
7.233TYPES::FIELD_PARAMETER_SET_TYPE Struct Reference	1100
7.233.1 Detailed Description	1100
7.233.2 Member Data Documentation	1100
7.233.2.1 PARAMETERS	1100
7.233.2.2 SET_INDEX	1100
7.233.2.3 SET_TYPE	1100
7.234TYPES::FIELD_PARAMETER_SETS_TYPE Struct Reference	1102
7.234.1 Detailed Description	1102
7.234.2 Member Data Documentation	1102
7.234.2.1 FIELD	1102
7.234.2.2 NUMBER_OF_PARAMETER_SETS	1102
7.234.2.3 PARAMETER_SETS	1103
7.234.2.4 SET_TYPE	1103
7.235TYPES::FIELD_PTR_TYPE Struct Reference	1104
7.235.1 Detailed Description	1104
7.235.2 Member Data Documentation	1104
7.235.2.1 PTR	1104
7.236TYPES::FIELD_SCALING_TYPE Struct Reference	1105
7.236.1 Detailed Description	1105
7.236.2 Member Data Documentation	1105
7.236.2.1 MAX_NUMBER_OF_DERIVATIVES	1105
7.236.2.2 MAX_NUMBER_OF_ELEMENT_PARAMETERS	1105
7.236.2.3 MESH_COMPONENT_NUMBER	1105
7.236.2.4 SCALE_FACTORS	1106
7.237TYPES::FIELD_SCALINGS_TYPE Struct Reference	1107
7.237.1 Detailed Description	1107
7.237.2 Member Data Documentation	1107
7.237.2.1 NUMBER_OF_SCALING_INDICES	1107
7.237.2.2 SCALING_TYPE	1107
7.237.2.3 SCALINGS	1107
7.238TYPES::FIELD_TYPE Struct Reference	1108
7.238.1 Detailed Description	1109
7.238.2 Member Data Documentation	1109
7.238.2.1 DECOMPOSITION	1109
7.238.2.2 DEPENDENT_TYPE	1109

7.238.2.3 DIMENSION	1109
7.238.2.4 FIELD_FINISHED	1110
7.238.2.5 FIELDS	1110
7.238.2.6 GEOMETRIC_FIELD	1110
7.238.2.7 GEOMETRIC_FIELD_PARAMETERS	1110
7.238.2.8 GLOBAL_NUMBER	1110
7.238.2.9 MAPPINGS	1110
7.238.2.10 NUMBER_OF_VARIABLES	1110
7.238.2.11 IPARAMETER_SETS	1110
7.238.2.12 REGION	1111
7.238.2.13 SCALINGS	1111
7.238.2.14 TYPE	1111
7.238.2.15 USER_NUMBER	1111
7.238.2.16 VARIABLE_TYPE_MAP	1111
7.238.2.17 VARIABLES	1111
7.239 TYPES::FIELD_VARIABLE_COMPONENT_TYPE Struct Reference	1112
7.239.1 Detailed Description	1112
7.239.2 Member Data Documentation	1113
7.239.2.1 COMPONENT_NUMBER	1113
7.239.2.2 DOMAIN	1113
7.239.2.3 FIELD	1113
7.239.2.4 FIELD_VARIABLE	1113
7.239.2.5 INTERPOLATION_TYPE	1113
7.239.2.6 MAX_NUMBER_OF_INTERPOLATION_PARAMETERS	1113
7.239.2.7 MESH_COMPONENT_NUMBER	1113
7.239.2.8 PARAM_TO_DOF_MAP	1114
7.239.2.9 REGION	1114
7.239.2.10 SCALING_INDEX	1114
7.240 TYPES::FIELD_VARIABLE_PTR_TYPE Struct Reference	1115
7.240.1 Detailed Description	1115
7.240.2 Member Data Documentation	1115
7.240.2.1 PTR	1115
7.241 TYPES::FIELD_VARIABLE_TYPE Struct Reference	1116
7.241.1 Detailed Description	1117
7.241.2 Member Data Documentation	1117
7.241.2.1 COMPONENTS	1117

7.241.2.2 DOF_LIST	1117
7.241.2.3 DOMAIN_MAPPING	1117
7.241.2.4 FIELD	1117
7.241.2.5 GLOBAL_DOF_LIST	1117
7.241.2.6 GLOBAL_DOF_OFFSET	1117
7.241.2.7 MAX_NUMBER_OF_INTERPOLATION_PARAMETERS	1118
7.241.2.8 NUMBER_OF_COMPONENTS	1118
7.241.2.9 NUMBER_OF_DOFS	1118
7.241.2.10 REGION	1118
7.241.2.11 TOTAL_NUMBER_OF_DOFS	1118
7.241.2.12 VARIABLE_NUMBER	1118
7.241.2.13 VARIABLE_TYPE	1118
7.242 TYPES::FIELDS_TYPE Struct Reference	1119
7.242.1 Detailed Description	1119
7.242.2 Member Data Documentation	1119
7.242.2.1 FIELDS	1119
7.242.2.2 NUMBER_OF_FIELDS	1119
7.242.2.3 REGION	1119
7.243 TYPES::GENERATED_MESH_REGULAR_TYPE Struct Reference	1120
7.243.1 Detailed Description	1120
7.243.2 Member Data Documentation	1120
7.243.2.1 BASIS	1120
7.243.2.2 GENERATED_MESH	1120
7.243.2.3 MAXIMUM_EXTENT	1121
7.243.2.4 MESH_DIMENSION	1121
7.243.2.5 NUMBER_OF_ELEMENTS_XI	1121
7.243.2.6 ORIGIN	1121
7.244 TYPES::GENERATED_MESH_TYPE Struct Reference	1122
7.244.1 Detailed Description	1122
7.244.2 Member Data Documentation	1122
7.244.2.1 GENERATED_TYPE	1122
7.244.2.2 MESH	1122
7.244.2.3 REGION	1122
7.244.2.4 REGULAR_MESH	1122
7.244.2.5 USER_NUMBER	1122
7.245 TYPES::LINEAR_DIRECT_SOLVER_TYPE Struct Reference	1123

7.245.1 Detailed Description	1123
7.245.2 Member Data Documentation	1123
7.245.2.1 DIRECT_SOLVER_TYPE	1123
7.245.2.2 LINEAR_SOLVER	1123
7.245.2.3 SOLVER_LIBRARY	1123
7.246TYPES::LINEAR_ITERATIVE_SOLVER_TYPE Struct Reference	1124
7.246.1 Detailed Description	1124
7.246.2 Member Data Documentation	1125
7.246.2.1 ABSOLUTE_TOLERANCE	1125
7.246.2.2 DIVERGENCE_TOLERANCE	1125
7.246.2.3 ITERATIVE_PRECONDITIONER_TYPE	1125
7.246.2.4 ITERATIVE_SOLVER_TYPE	1125
7.246.2.5 KSP	1125
7.246.2.6 LINEAR_SOLVER	1125
7.246.2.7 MAXIMUM_NUMBER_OF_ITERATIONS	1125
7.246.2.8 PC	1126
7.246.2.9 RELATIVE_TOLERANCE	1126
7.246.2.10 SOLVER_LIBRARY	1126
7.247TYPES::LINEAR_SOLVER_TYPE Struct Reference	1127
7.247.1 Detailed Description	1127
7.247.2 Member Data Documentation	1127
7.247.2.1 DIRECT_SOLVER	1127
7.247.2.2 ITERATIVE_SOLVER	1127
7.247.2.3 LINEAR_SOLVER_TYPE	1127
7.247.2.4 SOLVER	1128
7.248TYPES::MATRIX_TYPE Struct Reference	1129
7.248.1 Detailed Description	1130
7.248.2 Member Data Documentation	1130
7.248.2.1 COLUMN_INDICES	1130
7.248.2.2 DATA_DP	1130
7.248.2.3 DATA_INTG	1130
7.248.2.4 DATA_L	1130
7.248.2.5 DATA_SP	1131
7.248.2.6 DATA_TYPE	1131
7.248.2.7 ID	1131
7.248.2.8 M	1131

7.248.2.9 MATRIX_FINISHED	1131
7.248.2.10 MAX_M	1131
7.248.2.11 MAX_N	1131
7.248.2.12 MAXIMUM_COLUMN_INDICES_PER_ROW	1131
7.248.2.13 N	1132
7.248.2.14 NUMBER_NON_ZEROS	1132
7.248.2.15 ROW_INDICES	1132
7.248.2.16 SIZE	1132
7.248.2.17 STORAGE_TYPE	1132
7.249 TYPES::MESH_DOFS_TYPE Struct Reference	1133
7.249.1 Detailed Description	1133
7.249.2 Member Data Documentation	1133
7.249.2.1 MESH	1133
7.249.2.2 NUMBER_OF_DOFs	1133
7.250 TYPES::MESH_ELEMENT_TYPE Struct Reference	1134
7.250.1 Detailed Description	1134
7.250.2 Member Data Documentation	1134
7.250.2.1 ADJACENT_ELEMENTS	1134
7.250.2.2 BASIS	1135
7.250.2.3 GLOBAL_ELEMENT_NODES	1135
7.250.2.4 GLOBAL_NUMBER	1135
7.250.2.5 NUMBER_OF_ADJACENT_ELEMENTS	1135
7.250.2.6 USER_ELEMENT_NODES	1135
7.250.2.7 USER_NUMBER	1135
7.251 TYPES::MESH_ELEMENTS_TYPE Struct Reference	1136
7.251.1 Detailed Description	1136
7.251.2 Member Data Documentation	1136
7.251.2.1 ELEMENTS	1136
7.251.2.2 ELEMENTS_FINISHED	1136
7.251.2.3 MESH	1137
7.251.2.4 NUMBER_OF_ELEMENTS	1137
7.252 TYPES::MESH_NODE_TYPE Struct Reference	1138
7.252.1 Detailed Description	1138
7.252.2 Member Data Documentation	1138
7.252.2.1 DOF_INDEX	1138
7.252.2.2 GLOBAL_NUMBER	1138

7.252.2.3 NUMBER_OF_DERIVATIVES	1139
7.252.2.4 NUMBER_OF_SURROUNDING_ELEMENTS	1139
7.252.2.5 PARTIAL_DERIVATIVE_INDEX	1139
7.252.2.6 SURROUNDING_ELEMENTS	1139
7.252.2.7 USER_NUMBER	1139
7.253TYPES::MESH_NODES_TYPE Struct Reference	1140
7.253.1 Detailed Description	1140
7.253.2 Member Data Documentation	1140
7.253.2.1 MESH	1140
7.253.2.2 NODES	1140
7.253.2.3 NUMBER_OF_NODES	1140
7.254TYPES::MESH_PTR_TYPE Struct Reference	1141
7.254.1 Detailed Description	1141
7.254.2 Member Data Documentation	1141
7.254.2.1 PTR	1141
7.255TYPES::MESH_TOPOLOGY_PTR_TYPE Struct Reference	1142
7.255.1 Detailed Description	1142
7.255.2 Member Data Documentation	1142
7.255.2.1 PTR	1142
7.256TYPES::MESH_TOPOLOGY_TYPE Struct Reference	1143
7.256.1 Detailed Description	1143
7.256.2 Member Data Documentation	1143
7.256.2.1 DOFS	1143
7.256.2.2 ELEMENTS	1143
7.256.2.3 MESH	1143
7.256.2.4 MESH_COMPONENT_NUMBER	1144
7.256.2.5 NODES	1144
7.257TYPES::MESH_TYPE Struct Reference	1145
7.257.1 Detailed Description	1146
7.257.2 Member Data Documentation	1146
7.257.2.1 DECOMPOSITIONS	1146
7.257.2.2 EMBEDDED_MESHES	1146
7.257.2.3 EMBEDDING_MESH	1146
7.257.2.4 GLOBAL_NUMBER	1146
7.257.2.5 MESH_EMBEDDED	1146
7.257.2.6 MESH_FINISHED	1147

7.257.2.7 MESHES	1147
7.257.2.8 NUMBER_OF_COMPONENTS	1147
7.257.2.9 NUMBER_OF_DIMENSIONS	1147
7.257.2.10 NUMBER_OF_ELEMENTS	1147
7.257.2.11 NUMBER_OF_EMBEDDED_MESHES	1147
7.257.2.12 NUMBER_OF_FACES	1147
7.257.2.13 NUMBER_OF_LINES	1147
7.257.2.14 REGION	1147
7.257.2.15 TOPOLOGY	1148
7.257.2.16 USER_NUMBER	1148
7.258 TYPES::MESHES_TYPE Struct Reference	1149
7.258.1 Detailed Description	1149
7.258.2 Member Data Documentation	1149
7.258.2.1 MESHES	1149
7.258.2.2 NUMBER_OF_MESHES	1149
7.258.2.3 REGION	1149
7.259 TYPES::NODE_TYPE Struct Reference	1150
7.259.1 Detailed Description	1150
7.259.2 Member Data Documentation	1150
7.259.2.1 GLOBAL_NUMBER	1150
7.259.2.2 INITIAL_POSITION	1150
7.259.2.3 LABEL	1150
7.259.2.4 USER_NUMBER	1151
7.260 TYPES::NODES_TYPE Struct Reference	1152
7.260.1 Detailed Description	1152
7.260.2 Member Data Documentation	1152
7.260.2.1 NODE_TREE	1152
7.260.2.2 NODES	1152
7.260.2.3 NODES_FINISHED	1153
7.260.2.4 NUMBER_OF_NODES	1153
7.260.2.5 REGION	1153
7.261 TYPES::NONLINEAR_SOLVER_TYPE Struct Reference	1154
7.261.1 Detailed Description	1154
7.261.2 Member Data Documentation	1154
7.261.2.1 SOLVER	1154
7.261.2.2 SOLVER_LIBRARY	1154

7.262TYPES::PROBLEM_CONTROL_TYPE Struct Reference	1155
7.262.1 Detailed Description	1155
7.262.2 Member Data Documentation	1155
7.262.2.1 CONTROL_FINISHED	1155
7.262.2.2 CONTROL_TYPE	1155
7.262.2.3 PROBLEM	1155
7.263TYPES::PROBLEM_LINEAR_DATA_TYPE Struct Reference	1156
7.263.1 Detailed Description	1156
7.263.2 Member Data Documentation	1156
7.263.2.1 SOLUTION	1156
7.264TYPES::PROBLEM_NONLINEAR_DATA_TYPE Struct Reference	1157
7.264.1 Detailed Description	1157
7.264.2 Member Data Documentation	1157
7.264.2.1 NUMBER_OF_ITERATIONS	1157
7.264.2.2 SOLUTION	1157
7.265TYPES::PROBLEM_PTR_TYPE Struct Reference	1158
7.265.1 Detailed Description	1158
7.265.2 Member Data Documentation	1158
7.265.2.1 PTR	1158
7.266TYPES::PROBLEM_TIME_DATA_TYPE Struct Reference	1159
7.266.1 Detailed Description	1159
7.266.2 Member Data Documentation	1159
7.266.2.1 SOLUTION	1159
7.267TYPES::PROBLEM_TYPE Struct Reference	1160
7.267.1 Detailed Description	1160
7.267.2 Member Data Documentation	1160
7.267.2.1 CLASS	1160
7.267.2.2 CONTROL	1161
7.267.2.3 GLOBAL_NUMBER	1161
7.267.2.4 NUMBER_OF_SOLUTIONS	1161
7.267.2.5 PROBLEM_FINISHED	1161
7.267.2.6 PROBLEMS	1161
7.267.2.7 SOLUTIONS	1161
7.267.2.8 SUBTYPE	1161
7.267.2.9 TYPE	1161
7.267.2.10USER_NUMBER	1162

7.268TYPES::PROBLEMS_TYPE Struct Reference	1163
7.268.1 Detailed Description	1163
7.268.2 Member Data Documentation	1163
7.268.2.1 NUMBER_OF_PROBLEMS	1163
7.268.2.2 PROBLEMS	1163
7.269TYPES::QUADRATURE_SCHEME_PTR_TYPE Struct Reference	1164
7.269.1 Detailed Description	1164
7.269.2 Member Data Documentation	1164
7.269.2.1 PTR	1164
7.270TYPES::QUADRATURE_SCHEME_TYPE Struct Reference	1165
7.270.1 Detailed Description	1165
7.270.2 Member Data Documentation	1165
7.270.2.1 GAUSS_BASIS_FNS	1165
7.270.2.2 GAUSS_POSITIONS	1166
7.270.2.3 GAUSS_WEIGHTS	1166
7.270.2.4 GLOBAL_NUMBER	1166
7.270.2.5 NUMBER_OF_GAUSS	1166
7.270.2.6 QUADRATURE	1166
7.271TYPES::QUADRATURE_TYPE Struct Reference	1167
7.271.1 Detailed Description	1167
7.271.2 Member Data Documentation	1167
7.271.2.1 BASIS	1167
7.271.2.2 GAUSS_ORDER	1168
7.271.2.3 NUMBER_OF_GAUSS_XI	1168
7.271.2.4 NUMBER_OF_SCHEMES	1168
7.271.2.5 QUADRATURE_SCHEME_MAP	1168
7.271.2.6 SCHEMES	1168
7.271.2.7 TYPE	1168
7.272TYPES::REGION_PTR_TYPE Struct Reference	1169
7.272.1 Detailed Description	1169
7.272.2 Member Data Documentation	1169
7.272.2.1 PTR	1169
7.273TYPES::REGION_TYPE Struct Reference	1170
7.273.1 Detailed Description	1170
7.273.2 Member Data Documentation	1171
7.273.2.1 COORDINATE_SYSTEM	1171

7.273.2.2 EQUATIONS_SETS	1171
7.273.2.3 FIELDS	1171
7.273.2.4 LABEL	1171
7.273.2.5 MESHES	1171
7.273.2.6 NODES	1171
7.273.2.7 NUMBER_OF_SUB_REGIONS	1171
7.273.2.8 PARENT_REGION	1171
7.273.2.9 REGION_FINISHED	1172
7.273.2.10 SUB_REGIONS	1172
7.273.2.11 USER_NUMBER	1172
7.274 TYPES::SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE Struct Reference	1173
7.274.1 Detailed Description	1173
7.274.2 Member Data Documentation	1173
7.274.2.1 MATRIX_VARIABLE_TYPES	1173
7.275 TYPES::SOLUTION_MAPPING_TYPE Struct Reference	1174
7.275.1 Detailed Description	1175
7.275.2 Member Data Documentation	1175
7.275.2.1 CREATE_VALUES_CACHE	1175
7.275.2.2 EQUATIONS_SET_TO_SOLVER_MAP	1175
7.275.2.3 EQUATIONS_SETS	1175
7.275.2.4 NUMBER_OF_EQUATIONS_SETS	1175
7.275.2.5 NUMBER_OF_ROWS	1175
7.275.2.6 NUMBER_OF_SOLVER_MATRICES	1175
7.275.2.7 ROW_DOFS_MAPPING	1176
7.275.2.8 SOLUTION	1176
7.275.2.9 SOLUTION_MAPPING_FINISHED	1176
7.275.2.10 SOLVER_COL_TO_EQUATIONS_SETS_MAP	1176
7.275.2.11 SOLVER_ROW_TO_EQUATIONS_SET_MAPS	1176
7.275.2.12 TOTAL_NUMBER_OF_ROWS	1176
7.276 TYPES::SOLUTION_PTR_TYPE Struct Reference	1177
7.276.1 Detailed Description	1177
7.276.2 Member Data Documentation	1177
7.276.2.1 PTR	1177
7.277 TYPES::SOLUTION_TYPE Struct Reference	1178
7.277.1 Detailed Description	1178
7.277.2 Member Data Documentation	1178

7.277.2.1 EQUATIONS_SET_ADDED_INDEX	1178
7.277.2.2 EQUATIONS_SET_TO_ADD	1178
7.277.2.3 PROBLEM	1179
7.277.2.4 SOLUTION_FINISHED	1179
7.277.2.5 SOLUTION_MAPPING	1179
7.277.2.6 SOLUTION_NUMBER	1179
7.277.2.7 SOLVER	1179
7.278TYPES::SOLVER_COL_TO_EQUATIONS_MAP_TYPE Struct Reference	1180
7.278.1 Detailed Description	1180
7.278.2 Member Data Documentation	1180
7.278.2.1 COUPLING_COEFFICIENTS	1180
7.278.2.2 EQUATIONS_COL_NUMBERS	1180
7.278.2.3 EQUATIONS_MATRIX_NUMBERS	1180
7.278.2.4 NUMBER_OF_EQUATIONS_MATRICES	1181
7.279TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE Struct Reference	1182
7.279.1 Detailed Description	1182
7.279.2 Member Data Documentation	1182
7.279.2.1 EQUATIONS	1182
7.279.2.2 SOLVER_COL_TO_EQUATIONS_MAPS	1182
7.280TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE Struct Reference	1183
7.280.1 Detailed Description	1183
7.280.2 Member Data Documentation	1184
7.280.2.1 COLUMN_DOFS_MAPPING	1184
7.280.2.2 NUMBER_OF_COLUMNS	1184
7.280.2.3 NUMBER_OF_DOFs	1184
7.280.2.4 SOLUTION_MAPPING	1184
7.280.2.5 SOLVER_COL_TO_EQUATIONS_SET_MAPS	1184
7.280.2.6 SOLVER_COL_TO_VARIABLE_MAPS	1184
7.280.2.7 SOLVER_MATRIX	1184
7.280.2.8 SOLVER_MATRIX_NUMBER	1185
7.280.2.9 TOTAL_NUMBER_OF_DOFs	1185
7.281TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE Struct Reference	1186
7.281.1 Detailed Description	1186
7.281.2 Member Data Documentation	1186
7.281.2.1 ADDITIVE_CONSTANT	1186
7.281.2.2 EQUATIONS_SET_INDICES	1187

7.281.2.3 NUMBER_OF_EQUATIONS_SETS	1187
7.281.2.4 VARIABLE	1187
7.281.2.5 VARIABLE_COEFFICIENT	1187
7.281.2.6 VARIABLE_DOF	1187
7.282TYPES::SOLVER_MATRICES_TYPE Struct Reference	1188
7.282.1 Detailed Description	1188
7.282.2 Member Data Documentation	1188
7.282.2.1 LIBRARY_TYPE	1188
7.282.2.2 MATRICES	1189
7.282.2.3 NUMBER_OF_MATRICES	1189
7.282.2.4 NUMBER_OF_ROWS	1189
7.282.2.5 RHS_VECTOR	1189
7.282.2.6 SOLUTION_MAPPING	1189
7.282.2.7 SOLVER	1189
7.282.2.8 SOLVER_MATRICES_FINISHED	1189
7.282.2.9 TOTAL_NUMBER_OF_ROWS	1189
7.282.2.10UPDATE_RHS_VECTOR	1190
7.283TYPES::SOLVER_MATRIX_PTR_TYPE Struct Reference	1191
7.283.1 Detailed Description	1191
7.283.2 Member Data Documentation	1191
7.283.2.1 PTR	1191
7.284TYPES::SOLVER_MATRIX_TYPE Struct Reference	1192
7.284.1 Detailed Description	1192
7.284.2 Member Data Documentation	1192
7.284.2.1 MATRIX	1192
7.284.2.2 MATRIX_NUMBER	1192
7.284.2.3 NUMBER_OF_COLUMNS	1193
7.284.2.4 SOLVER_MATRICES	1193
7.284.2.5 SOLVER_VECTOR	1193
7.284.2.6 STORAGE_TYPE	1193
7.284.2.7 UPDATE_MATRIX	1193
7.285TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE Struct Reference	1194
7.285.1 Detailed Description	1194
7.285.2 Member Data Documentation	1194
7.285.2.1 COUPLING_COEFFICIENTS	1194
7.285.2.2 EQUATIONS_ROW_NUMBER	1194

7.285.2.3 EQUATIONS_SET	1194
7.285.2.4 NUMBER_OF_ROWS	1195
7.286TYPES::SOLVER_TYPE Struct Reference	1196
7.286.1 Detailed Description	1196
7.286.2 Member Data Documentation	1197
7.286.2.1 EIGENPROBLEM_SOLVER	1197
7.286.2.2 LINEAR_SOLVER	1197
7.286.2.3 NONLINEAR_SOLVER	1197
7.286.2.4 OUTPUT_TYPE	1197
7.286.2.5 SOLUTION	1197
7.286.2.6 SOLUTION_MAPPING	1197
7.286.2.7 SOLVE_TYPE	1197
7.286.2.8 SOLVER_FINISHED	1198
7.286.2.9 SOLVER_MATRICES	1198
7.286.2.10SPARSITY_TYPE	1198
7.286.2.11TIME_INTEGRATION_SOLVER	1198
7.287TYPES::SOURCE_EQUATIONS_MATRICES_MAP_TYPE Struct Reference	1199
7.287.1 Detailed Description	1199
7.287.2 Member Data Documentation	1199
7.287.2.1 EQUATIONS_ROW_TO_SOURCE_DOF_MAP	1199
7.287.2.2 SOURCE_DOF_TO_EQUATIONS_ROW_MAP	1199
7.288TYPES::TIME_INTEGRATION_SOLVER_TYPE Struct Reference	1200
7.288.1 Detailed Description	1200
7.288.2 Member Data Documentation	1200
7.288.2.1 SOLVER	1200
7.288.2.2 SOLVER_LIBRARY	1200
7.289TYPES::VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE Struct Reference	1201
7.289.1 Detailed Description	1201
7.289.2 Member Data Documentation	1201
7.289.2.1 COLUMN_DOF	1201
7.290TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE Struct Reference	1202
7.290.1 Detailed Description	1202
7.290.2 Member Data Documentation	1202
7.290.2.1 DOF_TO_COLUMNS_MAPS	1202
7.290.2.2 DOF_TO_ROWS_MAP	1203
7.290.2.3 EQUATIONS_MATRIX_NUMBERS	1203

7.290.2.4 NUMBER_OF_EQUATIONS_MATRICES	1203
7.290.2.5 VARIABLE	1203
7.290.2.6 VARIABLE_INDEX	1203
7.290.2.7 VARIABLE_TYPE	1203
7.291 TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE Struct Reference	1204
7.291.1 Detailed Description	1204
7.291.2 Member Data Documentation	1204
7.291.2.1 ADDITIVE_CONSTANTS	1204
7.291.2.2 COLUMN_NUMBERS	1204
7.291.2.3 COUPLING_COEFFICIENTS	1205
7.292 TYPES::VECTOR_TYPE Struct Reference	1206
7.292.1 Detailed Description	1206
7.292.2 Member Data Documentation	1206
7.292.2.1 DATA_DP	1206
7.292.2.2 DATA_INTG	1207
7.292.2.3 DATA_L	1207
7.292.2.4 DATA_SP	1207
7.292.2.5 DATA_TYPE	1207
7.292.2.6 ID	1207
7.292.2.7 N	1207
7.292.2.8 SIZE	1207
7.292.2.9 VECTOR_FINISHED	1207
8 File Documentation	1209
8.1 additional_doc/Installation.dox File Reference	1209
8.2 additional_doc/Test.dox File Reference	1210
8.3 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/base_routines.f90 File Reference . . .	1211
8.3.1 Detailed Description	1216
8.3.2 LICENSE	1216
8.4 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/basis_routines.f90 File Reference . . .	1217
8.5 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_c.c File Reference . . .	1218
8.5.1 Define Documentation	1218
8.5.1.1 CHARTYPE	1218
8.5.1.2 COMPLEXTYPE	1219
8.5.1.3 DOUBLECOMPLEXTYPE	1219
8.5.1.4 DOUBLETYPE	1219
8.5.1.5 FLIPENDIAN	1219

8.5.1.6	FLOATTYPE	1219
8.5.1.7	INTEGER TYPE	1219
8.5.1.8	LOGICAL TYPE	1219
8.5.1.9	LONGINTTYPE	1219
8.5.1.10	MAXBINFILES	1219
8.5.1.11	QUADRUPLECOMPLEXTYPE	1219
8.5.1.12	QUADRUPLETYP	1220
8.5.1.13	SAMEENDIAN	1220
8.5.1.14	SHORTINTTYPE	1220
8.5.2	Typedef Documentation	1220
8.5.2.1	logical	1220
8.5.3	Function Documentation	1220
8.5.3.1	BinaryCloseFile	1220
8.5.3.2	BinaryOpenFile	1220
8.5.3.3	BinaryReadFile	1220
8.5.3.4	BinarySetFile	1220
8.5.3.5	BinarySkipFile	1221
8.5.3.6	BinaryWriteFile	1221
8.5.3.7	IsBinaryFileOpen	1221
8.5.3.8	IsEndBinaryFile	1221
8.5.4	Variable Documentation	1221
8.5.4.1	binaryfiles	1221
8.6	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_f.f90 File Reference	1222
8.6.1	Detailed Description	1224
8.6.2	LICENSE	1224
8.7	d:/Users/tyu011/workspace/OpenCMISS-trunk/srcblas.f90 File Reference	1226
8.7.1	Detailed Description	1226
8.7.2	LICENSE	1226
8.8	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/classical_field_routines.f90 File Reference	1228
8.8.1	Detailed Description	1228
8.8.2	LICENSE	1229
8.9	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss.f90 File Reference	1230
8.9.1	Detailed Description	1230
8.10	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_mpi.f90 File Reference	1231
8.10.1	Detailed Description	1231
8.10.2	LICENSE	1231

8.11 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_parmetis.f90 File Reference . . .	1233
8.11.1 Detailed Description	1233
8.11.2 LICENSE	1233
8.12 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_petsc.f90 File Reference . . .	1235
8.12.1 Detailed Description	1241
8.12.2 LICENSE	1241
8.13 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_petsc_types.f90 File Reference . . .	1242
8.13.1 Detailed Description	1242
8.13.2 LICENSE	1242
8.14 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/computational_environment.f90 File Reference	1244
8.14.1 Detailed Description	1245
8.14.2 LICENSE	1245
8.15 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/constants.f90 File Reference	1247
8.16 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/coordinate_routines.f90 File Reference	1248
8.16.1 Detailed Description	1251
8.16.2 LICENSE	1251
8.17 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/distributed_matrix_vector.f90 File Reference	1253
8.18 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/domain_mappings.f90 File Reference	1254
8.18.1 Detailed Description	1255
8.18.2 LICENSE	1255
8.19 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/elasticity_routines.f90 File Reference	1256
8.19.1 Detailed Description	1256
8.19.2 LICENSE	1256
8.20 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/electromechanics_routines.f90 File Reference	1257
8.20.1 Detailed Description	1257
8.20.2 LICENSE	1257
8.21 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_mapping_routines.f90 File Reference	1258
8.22 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_matrices_routines.f90 File Reference	1259
8.23 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_constants.f90 File Reference	1260
8.23.1 Detailed Description	1264
8.23.2 LICENSE	1264
8.24 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_routines.f90 File Reference	1265

8.24.1	Detailed Description	1271
8.24.2	LICENSE	1271
8.25	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_c.c File Reference	1272
8.25.1	Typedef Documentation	1272
8.25.1.1	integer	1272
8.25.1.2	logical	1272
8.25.2	Function Documentation	1272
8.25.2.1	CStringLen	1272
8.25.2.2	PackCharacters	1272
8.25.2.3	UnPackCharacters	1272
8.26	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_f.f90 File Reference	1273
8.26.1	Detailed Description	1273
8.26.2	LICENSE	1273
8.27	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_IO_routines.f90 File Reference	1275
8.27.1	Detailed Description	1279
8.27.2	LICENSE	1279
8.28	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_routines.f90 File Reference	1280
8.28.1	Detailed Description	1285
8.28.2	LICENSE	1285
8.29	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite_element_routines.f90 File Reference	1286
8.29.1	Detailed Description	1286
8.29.2	LICENSE	1286
8.30	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/fluid_mechanics_routines.f90 File Reference	1288
8.30.1	Detailed Description	1288
8.30.2	LICENSE	1288
8.31	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/generated_mesh_routines.f90 File Reference	1289
8.31.1	Detailed Description	1290
8.31.2	LICENSE	1290
8.32	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/input_output.f90 File Reference	1291
8.32.1	Detailed Description	1305
8.32.2	LICENSE	1305
8.33	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/iso_varying_string.f90 File Reference	1306
8.33.1	Detailed Description	1309
8.34	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/kinds.f90 File Reference	1310

8.34.1 Detailed Description	1311
8.34.2 LICENSE	1311
8.35 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lapack.f90 File Reference	1312
8.35.1 Detailed Description	1312
8.35.2 LICENSE	1312
8.36 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/Laplace_equations_routines.f90 File Reference	1314
8.36.1 Detailed Description	1314
8.36.2 LICENSE	1315
8.37 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lists.f90 File Reference	1316
8.37.1 Detailed Description	1320
8.37.2 LICENSE	1320
8.38 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_aix.f90 File Reference	1321
8.38.1 Detailed Description	1321
8.38.2 LICENSE	1322
8.39 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_irix.f90 File Reference	1323
8.39.1 Detailed Description	1323
8.39.2 LICENSE	1323
8.40 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_linux.f90 File Reference	1324
8.40.1 Detailed Description	1324
8.40.2 LICENSE	1324
8.41 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_vms.f90 File Reference	1325
8.41.1 Detailed Description	1325
8.41.2 LICENSE	1325
8.42 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_win32.f90 File Reference	1326
8.42.1 Detailed Description	1326
8.42.2 LICENSE	1326
8.43 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/math.f90 File Reference	1327
8.43.1 Detailed Description	1328
8.43.2 LICENSE	1328
8.44 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/matrix_vector.f90 File Reference	1330
8.44.1 Detailed Description	1339
8.44.2 LICENSE	1339

8.44.3 MATRIX STORAGE STRUCTURES	1340
8.44.3.1 COMPRESSED-ROW STORAGE:	1340
8.44.3.2 COMPRESSED-COLUMN STORAGE:	1340
8.44.3.3 ROW-COLUMN STORAGE:	1341
8.45 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/mesh_routines.f90 File Reference . . .	1342
8.45.1 Detailed Description	1348
8.45.2 LICENSE	1348
8.46 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/node_routines.f90 File Reference . . .	1349
8.46.1 Detailed Description	1349
8.46.2 LICENSE	1349
8.47 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_constants.f90 File Reference .	1351
8.47.1 Detailed Description	1353
8.47.2 LICENSE	1353
8.48 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_routines.f90 File Reference .	1355
8.48.1 Detailed Description	1358
8.48.2 LICENSE	1358
8.49 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/region_routines.f90 File Reference . .	1360
8.49.1 Detailed Description	1361
8.49.2 LICENSE	1361
8.50 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solution_mapping_routines.f90 File Reference	1362
8.51 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_matrices_routines.f90 File Reference	1363
8.51.1 Detailed Description	1364
8.51.2 LICENSE	1364
8.52 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_routines.f90 File Reference . . .	1365
8.52.1 Detailed Description	1371
8.52.2 LICENSE	1371
8.53 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/sorting.f90 File Reference	1372
8.53.1 Detailed Description	1372
8.53.2 LICENSE	1372
8.54 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/strings.f90 File Reference	1374
8.54.1 Detailed Description	1378
8.54.2 LICENSE	1378
8.55 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_c.c File Reference	1380
8.55.1 Define Documentation	1380
8.55.1.1 SYSTEM_CPU	1380

8.55.1.2	TOTAL_CPU	1380
8.55.1.3	USER_CPU	1380
8.55.2	Function Documentation	1380
8.55.2.1	CPUTimer	1380
8.56	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_f.f90 File Reference	1381
8.56.1	Detailed Description	1381
8.56.2	LICENSE	1381
8.57	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/trees.f90 File Reference	1383
8.57.1	Detailed Description	1385
8.57.2	LICENSE	1385
8.58	d:/Users/tyu011/workspace/OpenCMISS-trunk/src/types.f90 File Reference	1386
8.58.1	Detailed Description	1394
8.58.2	LICENSE	1394

Chapter 1

openCMISS Documentation

An open source interactive computer program for Continuum Mechanics, Image analysis, Signal processing and System Identification. Target usage: Bioengineering application of finite element analysis, boundary element and collocation techniques.

1.1 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Chapter 2

Obtaining the Code and Setting up the Development Environment

2.1 Obtaining the Code and Libraries

To obtain the openCMISS source you need to check it out from the subversion repository. There are two parts to openCMISS to obtain - openCMISS itself and the various libraries it needs. In your root openCMISS directory, make the opencmiss and opencmissextras directories.

2.1.1 Obtain the Code

The openCMISS repository is at <https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm>
To check out the main trunk of openCMISS issue the following command in the opencmiss directory:

```
svn co https://opencmiss.svn.sourceforge.net/svnroot/opencmiss/cm/trunk cm
```

If you are not familiar with subversion, have a look at <http://svnbook.red-bean.com>.

2.1.2 Obtain the Libraries

The openCMISS libraries repository is at <http://www.physiome.ox.ac.uk/svn/opencmissextras/cm>
To check out the main trunk of the various libraries required with openCMISS issue the following command in the opencmissextras directory:

```
svn co http://www.physiome.ox.ac.uk/svn/opencmissextras/cm/trunk/external/architecture cm/external/architec
```

where architecture is the appropriate architecture for the machine. Possible architectures are:

- i386-win32
- i386-win32-debug
- i686-linux
- i686-linux-debug
- x86_64-linux
- x86_64-linux-debug
- rs6000-32-aix
- rs6000-32-aix-debug

Currently, the svn repository for openCMISS libraries is down. An alternative location for the libraries within the ABI is on hpc. Go to \bioengsmb and copy the necessary files. The folder structure is the same as svn repository.

2.1.3 Makefile Structure

The top level makefile will eventually build a library. In the examples directory there are separate compilable "applications" with individual makefiles. However, the library stuff isn't there as we need to code the bindings.

2.2 Programmer documentation

This programmer documentation is written using DocBook with the source located in the opencmiss/cm/doc/programmer_documentation/ folder. The source XML code can be transformed into either chunked or combined documents in PDF or HTML format. The Makefile can be used to perform the transformation using make single or make chunk in the above folder. PDFs can be generated using make fo Doxygen can be included .

2.3 Project Setup

2.3.1 On AIX 5.3 (HPC)

2.3.1.1 Set environment

Set environment variable to point to openCMISS

```
setenv OPENCMISS_ROOT <path to your opencmiss folder>
```

or

```
export OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to openCMISS-extras

```
setenv OPENCMISSEXTRAS_ROOT <path to your opencmissextras folder>
```

or

```
export OPENCMISSEXTRAS_ROOT=<path to your opencmissextras folder>
```

2.3.1.2 Set MPI

Create a file called hostfile.list in your home directory. Inside the file, add several lines of "hpc.bioeng.auckland.ac.nz" In .rhost file in the home direcotry, add "hpc <username>"

2.3.1.3 Compile

Change directory to opencmiss/cm Change directory to examples/ Use gmake. This should result in a binary that you can run in the bin/rs6000-32-aix folder.

2.3.2 On Ubuntu 8.04

2.3.2.1 Set environment

Set environment variable to point to opencmiss

```
setenv OPENCMISS_ROOT <path to your opencmiss folder>
```

or

```
export OPENCMISS_ROOT=<path to your opencmiss folder>
```

Set environment variable to point to openCMISS-extras

```
setenv OPENCMISS_EXTRAS_ROOT <path to your opencmissextras folder>
```

or

```
export OPENCMISS_EXTRAS_ROOT=<path to your opencmissextras folder>
```

It is also helpful to add the following

```
setenv PATH ${OPENCMISS_EXTRAS_ROOT}/cm/external/${archname}/bin:${PATH}
setenv PATH ${OPENCMISS_ROOT}/cm/bin/${archname}:${PATH}
```

where \${archname} is the appropriate architecture e.g., i686-linux, x86_64-linux. If you are using totalview you will also need to add

```
setenv LM_LICENSE_FILES <path-to-the-flex-directory>
```

2.3.2.2 Install Compilers

Download Intel Fotran Compiler from here. Extract the file and follow the install.htm to install

2.3.2.3 Compile

To build an example project:

```
make
```

To run the example project:

```
mpd & mpirun -n 2 path/to/the/execution/file
```

To debug the project using TotalView:

```
mpd & mpirun -tv 2 path/to/the/execution/file
```

2.3.3 On Windows XP (Visual Studio 2005)

2.3.3.1 Install Compilers

Download Intel Fortran Compiler from here. Execute the exe file and follow the installation wizard.

2.3.3.2 Install MPI

Download MPICH2 from here. You can either download the source archive and follow the README.windows file to install or download the installer to install. Set bin folder to the path To start the MPI, run

```
smpd -start
```

in command window. NOTE: as from MPICH2 version 1.0.7 the library names have changed. libmpich2 has now become libmpi!

2.3.3.3 Compile and Debug

Build the Fortran project under the debug mode and generate the opencmissstest-debug.exe file. In the C Project (since the Fortran projects do not support MPI cluster debugger), configure the debugging properties according to this. The MPIShim location is in the path similar to C: Files Visual Studio 8 Debugger\x86\mpishim.exe. Debug the C project.

2.3.3.4 Run

To run the project in the command window:

```
mpiexec -n 2 -localroot <path to the execution file>
```

2.3.4 On Windows Vista (Visual Studio 2008)

Install Compilers Download Intel Fortran Compiler from here. Execute the exe file and follow the installation wizard.

2.3.4.1

Before you install MPICH2 under Vista you must turn off User Account Control

1. Goto Start -> Control Panel
2. Double-click on User Accounts
3. Click "Turn User Account Control on or off"
4. Untick "Use User Account Control (UAC) to help protect your computer" and click OK
5. Restart your computer.

Download from here. Choose the Win32 IA32 (binary) option. Run the downloaded .msi file. Follow all instructions and install "For everybody". Once you have installed MPICH2 you can turn User Account Control back on. Follow the instructions above and in 4. tick the "Use User Account Control ...". NOTE: as from MPICH2 version 1.0.7 the library names have changed. libmpich2 has now become libmpi!

2.3.4.2 Compile

For each example, go into the VisualopenCMiss_08 folder. Double click the VisualopenCMiss project solution file to lauch Visual Studio.

2.4 Libraries Build (Optional)

2.4.1 Compiling PETSc

Note this is assuming you have the Intel Fortran compiler version 10.1.024. Adjust the version string as necessary.

2.4.1.1 Step1: Linux Environment installation and Compiler Environment Set up (For Windows only)

Under windows system:

- Install Cygwin if you need to. Cywin can be found here. Make sure you include the make and python modules when you install.
- Launch a Command Prompt Window
- Run the ifortvars.bat batch file to setup your Intel Fortran environment. e.g., "C:\Program Files\Intel\Compiler\Fortran\10.1.024\IA32\Bin\ifortvars.bat"
- Run the Cygwin batch file to setup the unix environment e.g., "C:\Cygwin\Cygwin.bat"

N.B: PetSc uses X, so make sure in linux environment, libX11-dev package is installed. Also make sure blas and lapack (-dev) packages are installed.

2.4.1.2 Step2: Compile PETSc

For Windows

- Change to the opencmissextras PETSc directory e.g., if opencmissextras root is E: and we are compiling PETSC version petsc-2.3.3-p8 then "cd /cygwin/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8"
- If you have MPICH2 version 1.0.7 or greater edit the python/BuildSystem/config/packages/MPI.py file. Find the self.liblist_mpich line. After the line "[fmpich2.lib', 'mpich2.lib'],", add the line "[fmpich2.lib', 'mpi.lib'],".
- PETSC_DIR=/cygdrive/e/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8; export PETSC_DIR
- For a debug install issue the following commands

```
PETSC_ARCH=cygwin-c-debug; export PETSC_ARCH
config/configure.py --prefix=/cygdrive/e/opencmissextras/cm/external/i386-win32-debug --with-shared=no
PETSC_ARCH=cygwin-c-debug; export PETSC_ARCH
```

For a non-debug install issue the following commands

```
PETSC_ARCH=cygwin-c-opt; export PETSC_ARCH
config/configure.py --prefix=/cygdrive/e/opencmissextras/cm/external/i386-win32 --with-shared=no --with-pic
PETSC_ARCH=cygwin-c-opt; export PETSC_ARCH
```

- make -e all
- make -e install

For AIX5.3

- Change to the opencmissextras PETSc directory. e.g /people/tyu011/workspace/opencmissextras/cm/external/packages/PETSc/2.3.3-p8
- setenv PETSC_DIR /people/tyu011/workspace/opencmissextras/cm/external/packages/PETSc/petsc-2.3.3-p8
- In config directory, copy the file aix5.1.0.0.py and rename it to aix5.3.0.0.py
- For a debug install issue the following commands

```
setenv PETSC_ARCH aix5.3.0.0
config/aix5.3.0.0.py --prefix=/people/tyu011/workspace/opencmissextras/cm/external/rs6000-32-aix-debug
setenv PETSC_ARCH aix5.3.0.0
```

For a non-debug install issue the following commands

```
setenv PETSC_ARCH aix5.3.0.0
config/aix5.3.0.0.py --prefix=/people/tyu011/workspace/opencmissextras/cm/external/rs6000-32-aix --without-debug
setenv PETSC_ARCH aix5.3.0.0
```

- make -e all
- make -e install

Chapter 3

Simple Test

Normal text.

User defined paragraph:

Contents of the paragraph.

New paragraph under the same heading.

Note:

This note consists of two paragraphs. This is the first paragraph.

And this is the second paragraph.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

This module handles all field related routines.

Chapter 4

Todo List

Member TYPES::BASIS_TYPE::NUMBER_OF_NODES_XI CHANGE TO NUMBER_OF_NODES_XIC i.e., xi coordinate index not xi???

Class TYPES::COORDINATE_SYSTEM_TYPE Have a list of coordinate systems and have a pointer in the coordinate_system_type back to the regions that use them.

Member TYPES::DECOMPOSITION_ELEMENTS_TYPE::ELEMENTS Change this to allocatable???

Member TYPES::DECOMPOSITION_TYPE::DOMAIN Change this to allocatable???

Member TYPES::DOMAIN_ELEMENTS_TYPE::ELEMENTS Change this to allocatable???

Member TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION1 move this to the decomposition face type

Member TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION2 move this to the decomposition face type

Member TYPES::DOMAIN_NODE_TYPE::SURROUNDING_ELEMENTS Change this to allocatable.

Member TYPES::DOMAIN_NODES_TYPE::NODES Change this to allocatable???

Member TYPES::EQUATIONS_SET_TYPE::FIXED_CONDITIONS Change name to BOUNDARY_CONDITIONS???

Member TYPES::EQUATIONS_TYPE::OUTPUT_TYPE move to equations set???

Member TYPES::EQUATIONS_TYPE::SPARSITY_TYPE move to equations set???

Member TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::ELEMENT_PARAM2DOF_MAP
Allow for multiple element parameters per element.

Member TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NODE_PARAM2DOF_MAP Don't allocate too much memory if there are different numbers of derivatives for different nodes.

Member TYPES::FIELD_SCALING_TYPE::SCALE_FACTORS Make scale factors nodally based for now. Will have to revert to element based and extended to be a matrix to allow for a global derivative to be mapped onto many different element derivatives at the points that closes meshes or for inconsistent xi directions

Member TYPES::MESH_ELEMENTS_TYPE::ELEMENTS Should this be allocatable.

Member TYPES::MESH_NODE_TYPE::SURROUNDING_ELEMENTS Change this to allocatable.

Member TYPES::MESH_NODES_TYPE::NODES Should this be allocatable???

Member TYPES::NODES_TYPE::NODES Should this be allocatable?

Class TYPES::QUADRATURE_SCHEME_TYPE Also evaluate the product of the basis functions at gauss points for speed???

Class TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE rename solver col to be solver dof here

Member BASE_ROUTINES::BASE_ROUTINES_FINALISE(**ERR, ERROR,*)** Finish this routine and deallocate memory.

File binary_file.f90 Fix naming convention and update to current code standard.

Member EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START(EQUATIONS_SET, EQUATIONS_SET)
Should this return a pointer to the equations???

Member FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM(USER_NUMBER, FIELD_VARIABLE_NUMBER)
change variable number to variable type.

File generated_mesh_routines.f90 Move generated regular mesh from mesh routines and generalise.

File lists.f90 Fix up and have this module use the sorting module for sorts

Member MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(DOMAIN_MAPPINGS, ERR, ERROR,*)
pass in the nodes mapping

Member MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(DOMAIN_MAPPINGS, ERR, ERROR,*)
finalise on error

Member MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(TOPOLOGY, ERR, ERROR,*)
pass in the dofs topolgy

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on exit

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in the domain elements

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(DOMAIN, ERR, ERROR,*)**
pass in domain topology

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE(DOMAIN, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in domain lines

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in domain nodes

Member **MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::MESH_CREATE_FINISH(REGION, MESH, ERR, ERROR,*)**
remove region

Member **MESH_ROUTINES::MESH_CREATE_REGULAR(USER_NUMBER, REGION, ORIGIN, MAXIMUM_EX)**
move to generated mesh routines

Member **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_FINALISE(TOPOLOGY, ERR, ERROR,*)**
pass in dofs

Member **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(TOPOLOGY, ERR, ERROR,*)**
finalise on error

Member **MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET(GLOBAL_NUMBER, ELEMENTS, E)**
use user number.

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(TOPOLOGY, ERR, ERROR,*)
as this is a user routine it should take a mesh pointer like create start and finish? Split this into destroy and finalise?

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(GLOBAL_NUMBER, ELEMENT_BASIS, COUNT, ERR, ERROR,*)
should take user number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(GLOBAL_NUMBER, ELEMENT_NODES, COUNT, ERR, ERROR,*)
specify by user number not global number

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(TOPOLOGY, ERR, ERROR,*)
pass in elements

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(TOPOLOGY, ERR, ERROR,*)
finalise on error

Member MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(GLOBAL_NUMBER, USER_NUMBER, COUNT, ERR, ERROR,*)
Check that the user number doesn't already exist.

Member MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(MESH, ERR, ERROR,*) pass in
the mesh topology

Member MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(MESH, ERR, ERROR,*)
finalise on error

Member MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE(TOPOLOGY, ERR, ERROR,*)
pass in nodes

Member MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(TOPOLOGY, ERR, ERROR,*)
finalise on errors

Member MESH_ROUTINES::MESSES_FINALISE(REGION, ERR, ERROR,*) pass in meshes

Member MESH_ROUTINES::MESSES_INITIALISE(REGION, ERR, ERROR,*) finalise on error

Member PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(PROBLEM, ERR, ERROR,*)
Should this return a pointer to the problem solution???

Group FIELD_ROUTINES_VariableTypes sort out variable access routines so that you are always accessing by variable type rather than variable number.

Group FIELD_ROUTINES_ParameterSetTypes make program defined constants negative?

Chapter 5

Module Documentation

5.1 BASE_ROUTINES::OutputType

Output type parameter.

Variables

- INTEGER(INTG), parameter [BASE_ROUTINES::GENERAL_OUTPUT_TYPE = 1](#)
General output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE = 2](#)
Diagnostic output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::TIMING_OUTPUT_TYPE = 3](#)
Timing output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::ERROR_OUTPUT_TYPE = 4](#)
Error output type.
- INTEGER(INTG), parameter [BASE_ROUTINES::HELP_OUTPUT_TYPE = 5](#)
Help output type.

5.1.1 Detailed Description

Output type parameter.

See also:

[BASE_ROUTINES](#)

5.1.2 Variable Documentation

5.1.2.1 INTEGER(INTG), parameter [BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE = 2](#)

Diagnostic output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 64 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), NODE_ROUTINES::NODES_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), REGION_ROUTINES::REGION_CREATE_FINISH(), REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

5.1.2.2 INTEGER(INTG),parameter BASE_ROUTINES::ERROR_OUTPUT_TYPE = 4

Error output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 66 of file base_routines.f90.

5.1.2.3 INTEGER(INTG),parameter BASE_ROUTINES::GENERAL_OUTPUT_TYPE = 1

General output type.

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES](#)

Definition at line 63 of file base_routines.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.1.2.4 INTEGER(INTG),parameter BASE_ROUTINES::HELP_OUTPUT_TYPE = 5

Help output type.

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES](#)

Definition at line 67 of file base_routines.f90.

5.1.2.5 INTEGER(INTG),parameter BASE_ROUTINES::TIMING_OUTPUT_TYPE = 3

Timing output type.

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES](#)

Definition at line 65 of file base_routines.f90.

5.2 BASE_ROUTINES::FileUnits

File unit parameters.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::ECHO_FILE_UNIT** = 10

File unit for echo files.

- INTEGER(INTG), parameter **BASE_ROUTINES::DIAGNOSTICS_FILE_UNIT** = 11

File unit for diagnostic files.

- INTEGER(INTG), parameter **BASE_ROUTINES::TIMING_FILE_UNIT** = 12

File unit for timing files.

- INTEGER(INTG), parameter **BASE_ROUTINES::LEARN_FILE_UNIT** = 13

File unit for learn files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO1_FILE_UNIT** = 21

File unit for general IO 1 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO2_FILE_UNIT** = 22

File unit for general IO 2 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO3_FILE_UNIT** = 23

File unit for general IO 3 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO4_FILE_UNIT** = 24

File unit for general IO 4 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::IO5_FILE_UNIT** = 25

File unit for general IO 5 files.

- INTEGER(INTG), parameter **BASE_ROUTINES::TEMPORARY_FILE_UNIT** = 80

File unit for temporary files.

- INTEGER(INTG), parameter **BASE_ROUTINES::OPEN_COMFILE_UNIT** = 90

File unit for open command files.

- INTEGER(INTG), parameter **BASE_ROUTINES::START_READ_COMFILE_UNIT** = 90

First file unit for read command files.

- INTEGER(INTG), parameter **BASE_ROUTINES::STOP_READ_COMFILE_UNIT** = 99

Last file unit for read command files.

5.2.1 Detailed Description

File unit parameters.

See also:

[BASE_ROUTINES](#)

5.2.2 Variable Documentation

5.2.2.1 INTEGER(INTG),parameter BASE_ROUTINES::DIAGNOSTICS_FILE_UNIT = 11

File unit for diagnostic files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 75 of file base_routines.f90.

5.2.2.2 INTEGER(INTG),parameter BASE_ROUTINES::ECHO_FILE_UNIT = 10

File unit for echo files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 74 of file base_routines.f90.

5.2.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IO1_FILE_UNIT = 21

File unit for general IO 1 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 78 of file base_routines.f90.

5.2.2.4 INTEGER(INTG),parameter BASE_ROUTINES::IO2_FILE_UNIT = 22

File unit for general IO 2 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 79 of file base_routines.f90.

5.2.2.5 INTEGER(INTG),parameter BASE_ROUTINES::IO3_FILE_UNIT = 23

File unit for general IO 3 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 80 of file base_routines.f90.

5.2.2.6 INTEGER(INTG),parameter BASE_ROUTINES::IO4_FILE_UNIT = 24

File unit for general IO 4 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 81 of file base_routines.f90.

5.2.2.7 INTEGER(INTG),parameter BASE_ROUTINES::IO5_FILE_UNIT = 25

File unit for general IO 5 files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 82 of file base_routines.f90.

5.2.2.8 INTEGER(INTG),parameter BASE_ROUTINES::LEARN_FILE_UNIT = 13

File unit for learn files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 77 of file base_routines.f90.

5.2.2.9 INTEGER(INTG),parameter BASE_ROUTINES::OPEN_COMFILE_UNIT = 90

File unit for open command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 84 of file base_routines.f90.

5.2.2.10 INTEGER(INTG),parameter BASE_ROUTINES::START_READ_COMFILE_UNIT = 90

First file unit for read command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 85 of file base_routines.f90.

5.2.2.11 INTEGER(INTG),parameter BASE_ROUTINES::STOP_READ_COMFILE_UNIT = 99

Last file unit for read command files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 86 of file base_routines.f90.

5.2.2.12 INTEGER(INTG),parameter BASE_ROUTINES::TEMPORARY_FILE_UNIT = 80

File unit for temporary files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 83 of file base_routines.f90.

5.2.2.13 INTEGER(INTG),parameter BASE_ROUTINES::TIMING_FILE_UNIT = 12

File unit for timing files.

See also:

[BASE_ROUTINES::FileUnits](#),[BASE_ROUTINES](#)

Definition at line 76 of file base_routines.f90.

5.3 BASE_ROUTINES::DiagnosticTypes

Diganostic type parameters.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::ALL_DIAG_TYPE = 1**
Type for setting diagnostic output in all routines.
- INTEGER(INTG), parameter **BASE_ROUTINES::IN_DIAG_TYPE = 2**
Type for setting diagnostic output in one routine.
- INTEGER(INTG), parameter **BASE_ROUTINES::FROM_DIAG_TYPE = 3**
Type for setting diagnostic output from one routine downwards.

5.3.1 Detailed Description

Diganostic type parameters.

See also:

[BASE_ROUTINES](#)

5.3.2 Variable Documentation

5.3.2.1 INTEGER(INTG),parameter **BASE_ROUTINES::ALL_DIAG_TYPE = 1**

Type for setting diagnostic output in all routines.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 93 of file base_routines.f90.

5.3.2.2 INTEGER(INTG),parameter **BASE_ROUTINES::FROM_DIAG_TYPE = 3**

Type for setting diagnostic output from one routine downwards.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 95 of file base_routines.f90.

5.3.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IN_DIAG_TYPE = 2

Type for setting diagnostic output in one routine.

See also:

[BASE_ROUTINES::DiagnosticTypes](#),[BASE_ROUTINES](#)

Definition at line 94 of file base_routines.f90.

5.4 BASE_ROUTINES::TimingTypes

Timing type parameters.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::ALL_TIMING_TYPE = 1**
Type for setting timing output in all routines.
- INTEGER(INTG), parameter **BASE_ROUTINES::IN_TIMING_TYPE = 2**
Type for setting timing output in one routine.
- INTEGER(INTG), parameter **BASE_ROUTINES::FROM_TIMING_TYPE = 3**
Type for setting timing output from one routine downwards.

5.4.1 Detailed Description

Timing type parameters.

See also:

[BASE_ROUTINES](#)

5.4.2 Variable Documentation

5.4.2.1 INTEGER(INTG),parameter **BASE_ROUTINES::ALL_TIMING_TYPE = 1**

Type for setting timing output in all routines.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 102 of file base_routines.f90.

5.4.2.2 INTEGER(INTG),parameter **BASE_ROUTINES::FROM_TIMING_TYPE = 3**

Type for setting timing output from one routine downwards.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 104 of file base_routines.f90.

5.4.2.3 INTEGER(INTG),parameter BASE_ROUTINES::IN_TIMING_TYPE = 2

Type for setting timing output in one routine.

See also:

[BASE_ROUTINES::TimingTypes](#),[BASE_ROUTINES](#)

Definition at line 103 of file base_routines.f90.

5.5 COORDINATE_ROUTINES::CoordinateSystemTypes

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE = 1](#)
Rectangular Cartesian coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_CYCLINDRICAL_POLAR_TYPE = 2](#)
Cylindrical polar coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_SPHERICAL_POLAR_TYPE = 3](#)
Spherical polar coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_PROLATE_SPHEROIDAL_TYPE = 4](#)
Prolate spheroidal coordinate system type.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_OBLATE_SPHEROIDAL_TYPE = 5](#)
Oblate spheroidal coordinate system type.

5.5.1 Detailed Description

See also:

[COORDINATE_ROUTINES](#) Coordinate system type parameters

5.5.2 Variable Documentation

5.5.2.1 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_CYCLINDRICAL_POLAR_TYPE = 2

Cylindrical polar coordinate system type.

See also:

[COORDINATE_ROUTINES](#)_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 68 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::CO\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP\(\)](#), [COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM\(\)](#), [COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST\(\)](#), [COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST\(\)](#), [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#), [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR\(\)](#), [COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE\(\)](#),

COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_OBLATE_SPHEROIDAL_TYPE = 5

Oblate spheroidal coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 71 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_PROLATE_SPHEROIDAL_TYPE = 4

Prolate spheroidal coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 70 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(),

COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_ROUTINES::DZX_DP().

5.5.2.4 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE = 1

Rectangular Cartesian coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 67 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE(), COORDINATE_ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), and MESH_ROUTINES::MESH_CREATE_REGULAR().

5.5.2.5 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_SPHERICAL_POLAR_TYPE = 3

Spherical polar coordinate system type.

See also:

COORDINATE_ROUTINES_CoordinateSystemTypes, [COORDINATE_ROUTINES](#)

Definition at line 69 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::CO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_

PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_-
CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_-
ROUTINES::D2ZX_DP(), COORDINATE_ROUTINES::DXZ_DP(), and COORDINATE_-
ROUTINES::DZX_DP().

5.6 COORDINATE_ROUTINES::RadialInterpolations

The type of radial interpolation for polar coordinate systems.

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_NO_RADIAL_INTERPOLATION_TYPE = 0](#)
No radial interpolation.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_INTERPOLATION_TYPE = 1](#)
r radial interpolation
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE = 2](#)
 r^2 radial interpolation
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE = 3](#)
 r^3 radial interpolation

5.6.1 Detailed Description

The type of radial interpolation for polar coordinate systems.

See also:

[COORDINATE_ROUTINES](#)

5.6.2 Variable Documentation

5.6.2.1 INTEGER(INTG),parameter [COORDINATE_ROUTINES::COORDINATE_NO_RADIAL_INTERPOLATION_TYPE = 0](#)

No radial interpolation.

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 78 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START\(\)](#), and [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR\(\)](#).

**5.6.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
CUBED_INTERPOLATION_TYPE = 3**

r^3 radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 81 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

**5.6.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
INTERPOLATION_TYPE = 1**

r radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 79 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

**5.6.2.4 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_RADIAL_-
SQUARED_INTERPOLATION_TYPE = 2**

r^2 radial interpolation

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#), [COORDINATE_ROUTINES](#)

Definition at line 80 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), and COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR().

5.7 COORDINATE_ROUTINES::JacobianType

The type of Jacobian to return when coordinate metrics are calculated.

Variables

- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_LINE_TYPE = 1](#)
Line type Jacobian.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_AREA_TYPE = 2](#)
Area type Jacobian.
- INTEGER(INTG), parameter [COORDINATE_ROUTINES::COORDINATE_JACOBIAN_VOLUME_TYPE = 3](#)
Volume type Jacobian.

5.7.1 Detailed Description

The type of Jacobian to return when coordinate metrics are calculated.

See also:

[COORDINATE_ROUTINES](#)

5.7.2 Variable Documentation

5.7.2.1 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_JACOBIAN_AREA_TYPE = 2

Area type Jacobian.

See also:

[COORDINATE_ROUTINES_JacobianTypes](#),[COORDINATE_ROUTINES](#)

Definition at line 89 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#).

5.7.2.2 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_JACOBIAN_LINE_TYPE = 1

Line type Jacobian.

See also:

[COORDINATE_ROUTINES_JacobianTypes](#),[COORDINATE_ROUTINES](#)

Definition at line 88 of file coordinate_routines.f90.

Referenced by [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE\(\)](#).

**5.7.2.3 INTEGER(INTG),parameter COORDINATE_ROUTINES::COORDINATE_-
JACOBIAN_VOLUME_TYPE = 3**

Volume type Jacobian.

See also:

COORDINATE_ROUTINES_JacobianTypes,[COORDINATE_ROUTINES](#)

Definition at line 90 of file coordinate_routines.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE().

5.8 DOMAIN_MAPPINGS::DomainType

Variables

- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL](#) = 1
The domain item is internal to the domain.
- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY](#) = 2
The domain item is on the boundary of the domain.
- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST](#) = 3
The domain item is ghosted from another domain.

5.8.1 Detailed Description

See also:

[DOMAIN_MAPPINGS](#)

5.8.2 Variable Documentation

5.8.2.1 INTEGER(INTG),parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY](#) = 2

The domain item is on the boundary of the domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#),[DOMAIN_MAPPINGS](#)

Definition at line 64 of file domain_mappings.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#).

5.8.2.2 INTEGER(INTG),parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST](#) = 3

The domain item is ghosted from another domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#),[DOMAIN_MAPPINGS](#)

Definition at line 65 of file domain_mappings.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS\(\)](#), and [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.8.2.3 INTEGER(INTG),parameter DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL = 1

The domain item is internal to the domain.

See also:

[DOMAIN_MAPPINGS::DomainType](#),[DOMAIN_MAPPINGS](#)

Definition at line 63 of file domain_mappings.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.9 EQUATIONS_SET_CONSTANTS::SetupTypes

Setup type parameters.

Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE = 1`
Initial setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE = 2`
Geometry setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE = 3`
Dependent variables.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE = 4`
Materials setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE = 5`
Source setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE = 6`
Source materials setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE = 7`
Analytic setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE = 8`
Fixed conditions.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUATIONS_TYPE = 9`
Equations setup.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINAL_TYPE = 9`
Final setup.

5.9.1 Detailed Description

Setup type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.9.2 Variable Documentation

5.9.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_ANALYTIC_TYPE = 7

Analytic setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 113 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-
FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_START(), and
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_-
SETUP().

5.9.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_DEPENDENT_TYPE = 3

Dependent variables.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 109 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_-
FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_-
START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
STANDARD_SETUP().

5.9.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- SETUP_EQUATIONS_TYPE = 9

Equations setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 115 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.9.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINAL_TYPE = 9

Final setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 116 of file equations_set_constants.f90.

5.9.2.5 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE = 8

Fixed conditions.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 114 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.9.2.6 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE = 2

Geometry setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 108 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.9.2.7 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE = 1

Initial setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 107 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_Set().

5.9.2.8 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE = 4

Materials setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 110 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.9.2.9 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE = 6

Source materials setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 112 of file equations_set_constants.f90.

5.9.2.10 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE = 5

Source setup.

See also:

[EQUATIONS_SET_CONSTANTS::SetupTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 111 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_FINISH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.10 EQUATIONS_SET_CONSTANTS::SetupActionTypes

Setup action type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION = 1](#)

Start setup action.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION = 2](#)

Finish setup action.

5.10.1 Detailed Description

Setup action type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.10.2 Variable Documentation

5.10.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION = 2](#)

Finish setup action.

See also:

[EQUATIONS_SET_CONSTANTS::SetupActionTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 124 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_FINISH\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_FINISH\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVARIANT_SETUP\(\)](#).

5.10.2.2 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION = 1](#)

Start setup action.

See also:

[EQUATIONS_SET_CONSTANTS::SetupActionTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 123 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_START(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET().

5.11 EQUATIONS_SET_CONSTANTS::FixedConditions

Fixed conditons parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED](#) = 0

The dof is not fixed.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION](#) = 1

The dof is fixed as a boundary condition.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION](#) = 2

The dof is set as a mixed boundary condition.

5.11.1 Detailed Description

Fixed conditons parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.11.2 Variable Documentation

5.11.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION](#) = 1

The dof is fixed as a boundary condition.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 132 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS\(\)](#), and [SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE\(\)](#).

5.11.2.2 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION](#) = 2

The dof is set as a mixed boundary condition.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 133 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.11.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED = 0

The dof is not fixed.

See also:

[EQUATIONS_SET_CONSTANTS::FixedConditions](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 131 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.12 EQUATIONS_SET_CONSTANTS::LinearityTypes

The problem linearity type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES = 3](#)

The number of problem linearity types defined.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR = 1](#)

The problem is linear.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR = 2](#)

The problem is non-linear.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR_BCS = 3](#)

The problem has non-linear boundary conditions.

5.12.1 Detailed Description

The problem linearity type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.12.2 Variable Documentation

5.12.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR = 1](#)

The problem is linear.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 141 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.12.2.2 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR = 2](#)

The problem is non-linear.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 142 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#).

5.12.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- NONLINEAR_BCS = 3

The problem has non-linear boundary conditions.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 143 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#).

5.12.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_- EQUATIONS_SET_LINEARITY_TYPES = 3

The number of problem linearity types defined.

See also:

[EQUATIONS_SET_CONSTANTS::LinearityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 140 of file equations_set_constants.f90.

5.13 EQUATIONS_SET_CONSTANTS::TimeDependenceTypes

The problem time dependence type parameters.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_TIME_TYPES = 3](#)

The number of problem time dependence types defined.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC = 1](#)

The problem is static and has no time dependence.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC = 2](#)

The problem is dynamic.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC = 3](#)

The problem is quasi-static.

5.13.1 Detailed Description

The problem time dependence type parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.13.2 Variable Documentation

5.13.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC = 2](#)

The problem is dynamic.

See also:

[EQUATIONS_SET_CONSTANTS_TimeDependenceTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 152 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#).

5.13.2.2 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC = 3](#)

The problem is quasi-static.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes, [EQUATIONS_SET_CONSTANTS](#)

Definition at line 153 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE().

5.13.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- STATIC = 1

The problem is static and has no time dependence.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes, [EQUATIONS_SET_CONSTANTS](#)

Definition at line 151 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE(), and LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.13.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_- EQUATIONS_SET_TIME_TYPES = 3

The number of problem time dependence types defined.

See also:

EQUATIONS_SET_CONSTANTS_TimeDependenceTypes, [EQUATIONS_SET_CONSTANTS](#)

Definition at line 150 of file equations_set_constants.f90.

5.14 EQUATIONS_SET_CONSTANTS::SolutionMethods

The solution method parameters.

Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_SOLUTION_METHOD = 6`
The number of solution methods defined.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM_SOLUTION_METHOD = 1`
Finite Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD = 2`
Boundary Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD_SOLUTION_METHOD = 3`
Finite Difference solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV_SOLUTION_METHOD = 4`
Finite Volume solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM_SOLUTION_METHOD = 5`
Grid-based Finite Element Method solution method.
- INTEGER(INTG), parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV_SOLUTION_METHOD = 6`
Grid-based Finite Volume solution method.

5.14.1 Detailed Description

The solution method parameters.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.14.2 Variable Documentation

5.14.2.1 INTEGER(INTG),parameter `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD = 2`

Boundary Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 162 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD_SOLUTION_METHOD = 3

Finite Difference solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 163 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM_SOLUTION_METHOD = 1

Finite Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 161 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV_SOLUTION_METHOD = 4

Finite Volume solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 164 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.5 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM_SOLUTION_METHOD = 5

Grid-based Finite Element Method solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 165 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.6 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV_SOLUTION_METHOD = 6

Grid-based Finite Volume solution method.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 166 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.14.2.7 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET SOLUTION_METHODS = 6

The number of solution methods defined.

See also:

[EQUATIONS_SET_CONSTANTS::SolutionMethods](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 160 of file equations_set_constants.f90.

5.15 EQUATIONS_SET_CONSTANTS::OutputTypes

Equations output types.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_OUTPUT = 0](#)

No output.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_TIMING_OUTPUT = 1](#)

Timing information output.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT = 2](#)

All below and equation matrices output.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT = 3](#)

All below and Element matrices output.

5.15.1 Detailed Description

Equations output types.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.15.2 Variable Documentation

5.15.2.1 INTEGER(INTG),parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT = 3](#)

All below and Element matrices output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#), [EQUATIONS_SET_CONSTANTS](#)

Definition at line 176 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET\(\)](#), and [EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE\(\)](#).

**5.15.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
MATRIX_OUTPUT = 2**

All below and equation matrices output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 175 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_-FEM(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET().

**5.15.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NO_OUTPUT = 0**

No output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 173 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET().

**5.15.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[EQUATIONS_SET_CONSTANTS::OutputTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 174 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_-FEM(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET().

5.16 EQUATIONS_SET_CONSTANTS::SparsityTypes

Equations matrices sparsity types.

Variables

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES = 1](#)

Use sparse matrices for the equations set.

- INTEGER(INTG), parameter [EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES = 2](#)

Use fully populated matrices for the equations set.

5.16.1 Detailed Description

Equations matrices sparsity types.

See also:

[EQUATIONS_SET_CONSTANTS](#)

5.16.2 Variable Documentation

5.16.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES = 2

Use fully populated matrices for the equations set.

See also:

[EQUATIONS_SET_CONSTANTS::SparsityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 184 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_TYPE_SET\(\)](#).

5.16.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES = 1

Use sparse matrices for the equations set.

See also:

[EQUATIONS_SET_CONSTANTS::SparsityTypes](#),[EQUATIONS_SET_CONSTANTS](#)

Definition at line 183 of file equations_set_constants.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_INITIALISE\(\)](#), and [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_TYPE_SET\(\)](#).

5.17 FIELD_ROUTINES::DependentTypes

Depedent field parameter types.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_INDEPENDENT_TYPE = 1](#)
Independent field type.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE = 2](#)
Dependent field type.

5.17.1 Detailed Description

Depedent field parameter types.

See also:

[FIELD_ROUTINES](#)

5.17.2 Variable Documentation

5.17.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE = 2](#)

Dependent field type.

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

Definition at line 72 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#), and [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP\(\)](#).

5.17.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_INDEPENDENT_TYPE = 1](#)

Independent field type.

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

Definition at line 71 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_CREATE_FINISH\(\)](#), [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR\(\)](#), and [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.18 FIELD_ROUTINES::DimensionTypes

Field dimension parameter types.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_SCALAR_DIMENSION_TYPE = 1](#)
Scalar field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_VECTOR_DIMENSION_TYPE = 2](#)
Vector field.

5.18.1 Detailed Description

Field dimension parameter types.

See also:

[FIELD_ROUTINES](#)

5.18.2 Variable Documentation

5.18.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_SCALAR_DIMENSION_TYPE = 1](#)

Scalar field.

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

Definition at line 79 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.18.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_VECTOR_DIMENSION_TYPE = 2](#)

Vector field.

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

Definition at line 80 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.19 FIELD_ROUTINES::FieldTypes

Field type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE](#) = 1
Geometric field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_FIBRE_TYPE](#) = 2
Fibre field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_GENERAL_TYPE](#) = 3
General field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_MATERIAL_TYPE](#) = 4
Material field.

5.19.1 Detailed Description

Field type parameters.

See also:

[FIELD_ROUTINES](#)

5.19.2 Variable Documentation

5.19.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_FIBRE_TYPE](#) = 2

Fibre field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 88 of file `field_routines.f90`.

Referenced by `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START()`, `FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE()`, `FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO()`, and `FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET()`.

5.19.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_GENERAL_TYPE](#) = 3

General field.

See also:

[FIELD_ROUTINES::FieldTypes](#),[FIELD_ROUTINES](#)

Definition at line 89 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_LABEL_FIELD_INFO_GET(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUIVATIONS_SET_STANDARD_SETUP().

5.19.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE = 1

Geometric field.

See also:

[FIELD_ROUTINES::FieldTypes](#), [FIELD_ROUTINES](#)

Definition at line 87 of file field_routines.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(), and FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET().

5.19.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_MATERIAL_TYPE = 4

Material field.

See also:

[FIELD_ROUTINES::FieldTypes](#), [FIELD_ROUTINES](#)

Definition at line 90 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_IO_ROUTINES::FIELD_LABEL_FIELD_INFO_GET().

5.20 FIELD_ROUTINES::InterpolationTypes

Field interpolation parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_CONSTANT_INTERPOLATION = 1](#)
Constant interpolation. One parameter for the field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ELEMENT_BASED_INTERPOLATION = 2](#)
Element based interpolation. Parameters are different in each element.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION = 3](#)
Node based interpolation. Parameters are nodal based and a basis function is used.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_POINT_BASED_INTERPOLATION = 4](#)
Point based interpolation. Parameters are different at each grid point.

5.20.1 Detailed Description

Field interpolation parameters.

See also:

[FIELD_ROUTINES](#)

5.20.2 Variable Documentation

5.20.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_CONSTANT_INTERPOLATION = 1](#)

Constant interpolation. One parameter for the field.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 97 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FI()`, `FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, and `FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE()`.

**5.20.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_ELEMENT_BASED_-
INTERPOLATION = 2**

Element based interpolation. Parameters are different in each element.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 98 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_-
COM(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_-
ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE().

**5.20.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NODE_BASED_-
INTERPOLATION = 3**

Node based interpolation. Parameters are nodal based and a basis function is used.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 99 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_-
MESH_COM(), FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD_-
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_ROUTINES::FIELD_-
VARIABLE_COMPONENT_INITIALISE(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_-
EQUATION_EQUATIONS_SET_STANDARD_SETUP().

**5.20.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_POINT_BASED_-
INTERPOLATION = 4**

Point based interpolation. Parameters are different at each grid point.

See also:

[FIELD_ROUTINES::InterpolationTypes](#),[FIELD_ROUTINES](#)

Definition at line 100 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FI(), FIELD_ROUTINES::FIELD_COMPONENT_MESH_-
COM(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), and FIELD_-
ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE().

5.21 FIELD_ROUTINES::VariableTypes

Field variable type parameters.

Variables

- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES = 4`
Number of different field variable types possible.
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE = 1`
Standard variable type i.e., u .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE = 2`
Normal derivative variable type i.e., du/dn .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE = 3`
First time derivative variable type i.e., du/dt .
- INTEGER(INTG), parameter `FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE = 4`
Second type derivative variable type i.e., d^2u/dt^2 .

5.21.1 Detailed Description

Field variable type parameters.

See also:

[FIELD_ROUTINES](#)

Todo

sort out variable access routines so that you are always accessing by variable type rather than variable number.

5.21.2 Variable Documentation

5.21.2.1 INTEGER(INTG),parameter `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE = 2`

Normal derivative variable type i.e., du/dn .

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 110 of file field_routines.f90.

Referenced by FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

5.21.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES = 4

Number of different field variable types possible.

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 108 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_FINISH(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE().

5.21.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE = 1

Standard variable type i.e., u.

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 109 of file field_routines.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_INTERPOLATION_SET(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_SET(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP().

5.21.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE = 3

First time derivative variable type i.e., du/dt.

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 111 of file field_routines.f90.

Referenced by FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET().

**5.21.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_TIME_DERIV2_-
VARIABLE_TYPE = 4**

Second type derivative variable type i.e., d^2u/dt^2 .

See also:

[FIELD_ROUTINES::VariableTypes](#),[FIELD_ROUTINES](#)

Definition at line 112 of file field_routines.f90.

Referenced by [FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET\(\)](#).

5.22 FIELD_ROUTINES::DofTypes

Field dof type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE = 1](#)
The dof is from a field variable component with constant interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE = 2](#)
The dof is from a field variable component with element based interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NODE_DOF_TYPE = 3](#)
The dof is from a field variable component with node based interpolation.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_POINT_DOF_TYPE = 4](#)
The dof is from a field variable component with point based interpolation.

5.22.1 Detailed Description

Field dof type parameters.

See also:

[FIELD_ROUTINES](#)

5.22.2 Variable Documentation

5.22.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE = 1](#)

The dof is from a field variable component with constant interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 119 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.22.2.2 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE = 2](#)

The dof is from a field variable component with element based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 120 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.22.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NODE_DOF_TYPE = 3

The dof is from a field variable component with node based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 121 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.22.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_POINT_DOF_TYPE = 4

The dof is from a field variable component with point based interpolation.

See also:

[FIELD_ROUTINES::DofTypes](#),[FIELD_ROUTINES](#)

Definition at line 122 of file field_routines.f90.

5.23 FIELD_ROUTINES::ParameterSetTypes

Field parameter set type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES = 99](#)
The maximum number of different parameter sets for a field.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_VALUES_SET_TYPE = 1](#)
The parameter set corresponding to the field values.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE = 2](#)
The parameter set corresponding to the field boundary conditions.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_INITIAL_CONDITIONS_SET_TYPE = 3](#)
The parameter set corresponding to the field initial conditions.

5.23.1 Detailed Description

Field parameter set type parameters.

[Todo](#)

make program defined constants negative?

See also:

[FIELD_ROUTINES](#)

5.23.2 Variable Documentation

5.23.2.1 INTEGER(INTG),parameter [FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE = 2](#)

The parameter set corresponding to the field boundary conditions.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 131 of file `field_routines.f90`.

Referenced by `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

5.23.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_INITIAL_CONDITIONS_SET_TYPE = 3

The parameter set corresponding to the field initial conditions.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 132 of file field_routines.f90.

5.23.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES = 99

The maximum number of different parameter sets for a field.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 129 of file field_routines.f90.

Referenced by [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#).

5.23.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_VALUES_SET_TYPE = 1

The parameter set corresponding to the field values.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#),[FIELD_ROUTINES](#)

Definition at line 130 of file field_routines.f90.

Referenced by [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE\(\)](#), [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY\(\)](#), [FIELD_ROUTINES::FIELD_CREATE_FINISH\(\)](#), [FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET\(\)](#), [FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS\(\)](#), [FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC\(\)](#), [FIELD_IO_ROUTINES::FIELD_IO_FILES_IMPORT\(\)](#), [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE\(\)](#), [SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE\(\)](#), and [SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE\(\)](#).

5.24 FIELD_ROUTINES::ScalingTypes

Field scaling type parameters.

Variables

- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_NO_SCALING = 0](#)
The field is not scaled.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_UNIT_SCALING = 1](#)
The field has unit scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ARC_LENGTH_SCALING = 2](#)
The field has arc length scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ARITHMETIC_MEAN_SCALING = 3](#)
The field has arithmetic mean of the arc length scaling.
- INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_HARMONIC_MEAN_SCALING = 4](#)
The field has geometric mean of the arc length scaling.

5.24.1 Detailed Description

Field scaling type parameters.

See also:

[FIELD_ROUTINES](#)

5.24.2 Variable Documentation

5.24.2.1 INTEGER(INTG), parameter [FIELD_ROUTINES::FIELD_ARC_LENGTH_SCALING = 2](#)

The field has arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#), [FIELD_ROUTINES](#)

Definition at line 141 of file `field_routines.f90`.

Referenced by `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

5.24.2.2 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_ARITHMETIC_MEAN_SCALING = 3

The field has arithmetic mean of the arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 142 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.24.2.3 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_HARMONIC_MEAN_SCALING = 4

The field has geometric mean of the arc length scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 143 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.24.2.4 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_NO_SCALING = 0

The field is not scaled.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 139 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.24.2.5 INTEGER(INTG),parameter FIELD_ROUTINES::FIELD_UNIT_SCALING = 1

The field has unit scaling.

See also:

[FIELD_ROUTINES::ScalingTypes](#),[FIELD_ROUTINES](#)

Definition at line 140 of file field_routines.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

5.25 GENERATED_MESH_ROUTINES::GeneratedMeshTypes

Generated mesh types.

Variables

- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_MESH_TYPE = 1](#)
A regular generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_POLAR_MESH_TYPE = 2](#)
A polar generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_FRACTAL_TREE_MESH_TYPE = 3](#)
A fractal tree generated mesh.

5.25.1 Detailed Description

Generated mesh types.

See also:

[GENERATED_MESH_ROUTINES](#)

5.25.2 Variable Documentation

5.25.2.1 INTEGER(INTG),parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_FRACTAL_TREE_MESH_TYPE = 3](#)

A fractal tree generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

Definition at line 69 of file generated_mesh_routines.f90.

5.25.2.2 INTEGER(INTG),parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_POLAR_MESH_TYPE = 2](#)

A polar generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

Definition at line 68 of file generated_mesh_routines.f90.

**5.25.2.3 INTEGER(INTG),parameter GENERATED_MESH_ROUTINES::GENERATED_-
MESH_REGULAR_MESH_TYPE = 1**

A regular generated mesh.

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

Definition at line 67 of file generated_mesh_routines.f90.

5.26 INPUT_OUTPUT::MatrixNameIndexFormat

Output type parameter.

Variables

- INTEGER(INTG), parameter [INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1](#)

Write the matrix name with out any indices.

- INTEGER(INTG), parameter [INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2](#)

Write the matrix name together with the matrix indices.

5.26.1 Detailed Description

Output type parameter.

See also:

[INPUT_OUTPUT](#)

5.26.2 Variable Documentation

5.26.2.1 INTEGER(INTG),parameter INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2

Write the matrix name together with the matrix indices.

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#),[INPUT_OUTPUT::MatrixNameIndexFormat](#)

Definition at line 63 of file input_output.f90.

Referenced by COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), MATRIX_VECTOR::MATRIX_OUTPUT(), [INPUT_OUTPUT::WRITE_STRING_MATRIX_DP\(\)](#), [INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG\(\)](#), [INPUT_OUTPUT::WRITE_STRING_MATRIX_L\(\)](#), [INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG\(\)](#), and [INPUT_OUTPUT::WRITE_STRING_MATRIX_SP\(\)](#).

5.26.2.2 INTEGER(INTG),parameter INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1

Write the matrix name with out any indices.

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#),[INPUT_OUTPUT::MatrixNameIndexFormat](#)

Definition at line 62 of file input_output.f90.

Referenced by INPUT_OUTPUT::WRITE_STRING_MATRIX_DPO(), INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_L(), INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG(), and INPUT_OUTPUT::WRITE_STRING_MATRIX_SP().

5.27 KINDS::IntegerKinds

Kind parameters for integer data types.

Variables

- INTEGER, parameter [KINDS::INTG](#) = SELECTED_INT_KIND(9)
Standard integer kind.
- INTEGER, parameter [KINDS::SINTG](#) = SELECTED_INT_KIND(4)
Short integer kind.
- INTEGER, parameter [KINDS::LINTG](#) = SELECTED_INT_KIND(18)
Long integer kind.
- INTEGER, parameter [KINDS::PTR](#) = INTG
Pointer integer kind.

5.27.1 Detailed Description

Kind parameters for integer data types.

See also:

[KINDS](#)

5.27.2 Variable Documentation

5.27.2.1 INTEGER,parameter KINDS::INTG = SELECTED_INT_KIND(9)

Standard integer kind.

See also:

[KINDS::IntegerKinds](#),[KINDS](#)

Definition at line 54 of file kinds.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and BINARY_-FILE::OPEN_CMISS_BINARY_FILE().

5.27.2.2 INTEGER,parameter KINDS::LINTG = SELECTED_INT_KIND(18)

Long integer kind.

See also:

[KINDS::IntegerKinds](#),[KINDS](#)

Definition at line 56 of file kinds.f90.

5.27.2.3 INTEGER,parameter KINDS::PTR = INTG

Pointer integer kind.

Definition at line 57 of file kinds.f90.

```
Referenced      by      COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_
FINISH(),      COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_NUMBER(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_PTR(),      COORDINATE-
ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(),      COORDINATE-
ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE-
SYSTEMS_INITIALISE(), MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH-
ROUTINES::DECOMPOSITION_CREATE_START(),      MESH_ROUTINES::DECOMPOSITION-
DESTROY(),      MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN-
CALCULATE(),      MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_SET(),
MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(),      DOMAIN-
MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(),      MESH-
ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(),      EQUATIONS_SET-
ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM(),      EQUATIONS-
SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE(),      EQUATIONS_SET-
ROUTINES::EQUATIONS_SET_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS-
SET_DESTROY(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT-
CALCULATE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_USER_NUMBER-
FIND(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FINALISE(),      FIELD-
ROUTINES::FIELD_CREATE_FINISH(),      FIELD_ROUTINES::FIELD_DESTROY(),      FIELD-
ROUTINES::FIELD_INTERPOLATE_GAUSS(),      FIELD_ROUTINES::FIELD_INTERPOLATE-
XI(),      FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT-
GET(),      FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(),      FIELD-
IO_ROUTINES::FIELD_IO_CREATE_FIELDS(),      FIELD_IO_ROUTINES::FIELD_IO-
ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS(),      FIELD_IO_ROUTINES::FIELD-
IO_ELEMENTAL_INFO_SET_FINALIZE(),      FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL-
INFO_SET_INITIALISE(),      FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET-
SORT(),      FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER-
FORTRAN(),      FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_(),      FIELD-
IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA(),      FIELD_IO-
ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(),      FIELD_IO_ROUTINES::FIELD-
IO_IMPORT_GLOBAL_MESH(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET-
ATTACH_LOCAL_PROCESS(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET-
FINALIZE(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE(),      FIELD-
IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT(),      FIELD_ROUTINES::FIELD-
VARIABLE_COMPONENT_INITIALISE(),      FIELD_ROUTINES::FIELDS_FINALISE(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE-
ELEMENT_CALCULATE(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION-
PROBLEM_STANDARD_SETUP(),      MESH_ROUTINES::MESH_CREATE_FINISH(),      MESH-
ROUTINES::MESH_CREATE_START(),      MESH_ROUTINES::MESH_DESTROY(),      MESH-
ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(),      MESH_ROUTINES::MESH-
NUMBER_OF_ELEMENTS_SET_PTR(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS-
ADJACENT_ELEMENTS_CALCULATE(),      MESH_ROUTINES::MESH_TOPOLOGY-
ELEMENTS_CREATE_FINISH(),      MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS-
CREATE_START(),      MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(),      MESH-
ROUTINES::MESH_TOPOLOGY_INITIALISE(),      MESH_ROUTINES::MESH_USER_NUMBER-
FIND(),      MESH_ROUTINES::MESSES_FINALISE(),      PROBLEM_ROUTINES::PROBLEM-
CREATE_FINISH(),      PROBLEM_ROUTINES::PROBLEM_CREATE_START(),      PROBLEM-
```

```
ROUTINES::PROBLEM_DESTROY_NUMBER(),           PROBLEM_ROUTINES::PROBLEM_-
DESTROY_PTR(),                  PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_-
SET_ADD(),          PROBLEM_ROUTINES::PROBLEM_SOLUTION_solve(),      PROBLEM_-
ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(),   PROBLEM_ROUTINES::PROBLEM_-
SOLUTIONS_FINALISE(),      PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(), 
PROBLEM_ROUTINES::PROBLEM_solve(),            PROBLEM_ROUTINES::PROBLEM_-
SOLVER_CREATE_START(),      PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(), 
PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(),        PROBLEM_ROUTINES::PROBLEM_-
USER_NUMBER_FIND(),         PROBLEM_ROUTINES::PROBLEMS_FINALISE(),       REGION_-
ROUTINES::REGION_CREATE_FINISH(),    REGION_ROUTINES::REGION_CREATE_START(), 
REGION_ROUTINES::REGION_DESTROY(),      REGION_ROUTINES::REGION_SUB_REGION_-
CREATE_FINISH(),      REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(), 
REGION_ROUTINES::REGION_USER_NUMBER_FIND(),   REGION_ROUTINES::REGION_-
USER_NUMBER_FIND_PTR(),      REGION_ROUTINES::REGIONS_FINALISE(),      SOLVER_-
ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(),        SOLVER_ROUTINES::SOLVER_-
LINEAR_ITERATIVE_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_LINEAR_-
ITERATIVE_SOLVE(),      SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(),  SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(),  SOLVER_MATRICES_-
ROUTINES::SOLVER_MATRICES_FINALISE(),      SOLVER_MATRICES_ROUTINES::SOLVER_-
MATRICES_INITIALISE(),      SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_-
OUTPUT(),          SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_-
SET(),          SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE(),  SOLVER_-
MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE(),  and SOLVER_-
ROUTINES::SOLVER_VARIABLES_UPDATE().
```

5.27.2.4 INTEGER,parameter KINDS::SINTG = SELECTED_INT_KIND(4)

Short integer kind.

See also:

[KINDS::IntegerKinds](#), [KINDS](#)

Definition at line 55 of file kinds.f90.

5.28 KINDS::RealKinds

Kind parameters for real data types.

Variables

- INTEGER, parameter **KINDS::SP** = SELECTED_REAL_KIND(6
Single precision real kind.
- INTEGER, parameter **KINDS::DP** = SELECTED_REAL_KIND(15
Double precision real kind.
- INTEGER, parameter **KINDS::QP** = SELECTED_REAL_KIND(30
Quadruple precision real kind.

5.28.1 Detailed Description

Kind parameters for real data types.

See also:

[KINDS](#)

5.28.2 Variable Documentation

5.28.2.1 INTEGER,parameter KINDS::DP = SELECTED_REAL_KIND(15

Double precision real kind.

See also:

[KINDS::RealKinds](#), [KINDS](#)

Definition at line 65 of file kinds.f90.

5.28.2.2 INTEGER,parameter KINDS::QP = SELECTED_REAL_KIND(30

Quadruple precision real kind.

See also:

[KINDS::RealKinds](#), [KINDS](#)

Definition at line 66 of file kinds.f90.

5.28.2.3 INTEGER,parameter KINDS::SP = SELECTED_REAL_KIND(6)

Single precision real kind.

See also:

[KINDS::RealKinds](#),[KINDS](#)

Definition at line 64 of file kinds.f90.

5.29 KINDS::ComplexKinds

Kind parameters for complex data types.

Variables

- INTEGER, parameter [KINDS::SPC](#) = KIND((1.0_SP)
- INTEGER, parameter [KINDS::_SP](#)

Single precision complex kind.

- INTEGER, parameter [KINDS::DPC](#) = KIND((1.0_DP)
- INTEGER, parameter [KINDS::_DP](#)

Double precision complex kind.

5.29.1 Detailed Description

Kind parameters for complex data types.

See also:

[KINDS](#)

5.29.2 Variable Documentation

5.29.2.1 INTEGER,parameter KINDS::_DP

Double precision complex kind.

See also:

[KINDS::ComplexKinds](#), [KINDS](#)

Definition at line 74 of file kinds.f90.

```
Referenced by      BASE_ROUTINES::BASE_ROUTINES_INITIALISE(),      COORDINATE_-
ROUTINES::CO(),      COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_-
RC_DP(),      COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), 
COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(),      COORDINATE_-
ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_-
INTERPOLATION_ADJUST(),      COORDINATE_ROUTINES::COORDINATE_METRICS_-
CALCULATE(),      COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START(), 
COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE(),      COORDINATE_-
ROUTINES::D2ZX_DP(),      MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_-
SET(),      MATHS::DETERMINANT_FULL_DP(),      COORDINATE_ROUTINES::DXZ_DP(), 
COORDINATE_ROUTINES::DZX_DP(), MATHS::EDP_DP(), MATHS::EIGENVALUE_FULL_DP(), 
MATHS::EIGENVECTOR_FULL_DP(), MATHS::EIGENVECTOR_FULL_SP(), EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM(), EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE(),      FIELD_ROUTINES::FIELD_-
INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-
METRICS_INITIALISE(),      FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_-
INITIALISE(),      FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(),
```

MATHS::I0_DP(), MATHS::I1_DP(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), MATHS::K0_DP(), MATHS::K1_DP(), MATHS::KDP_DP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP2(), MESH_ROUTINES::MESH_CREATE_REGULAR(), NODE_ROUTINES::NODES_CREATE_START(), STRINGS::NUMBER_TO_CHARACTER_DP(), STRINGS::NUMBER_TO_VSTRING_DP(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), and SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE().

5.29.2.2 INTEGER,parameter KINDS::SP

Single precision complex kind.

See also:

[KINDS::ComplexKinds](#), [KINDS](#)

Definition at line 73 of file kinds.f90.

Referenced by BASE_ROUTINES::BASE_ROUTINES_INITIALISE(), COORDINATE_ROUTINES::COO(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MATHS::DETERMINANT_FULL_SP(), MATHS::EDP_SP(), MATHS::EIGENVALUE_FULL_SP(), MATHS::EIGENVECTOR_FULL_SP(), BASE_ROUTINES::ENTERS(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM(), MATHS::I0_SP(), MATHS::I1_SP(), MATHS::INVERT_FULL_SP(), MATHS::K0_SP(), MATHS::K1_SP(), MATHS::KDP_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP2(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(), and BASE_ROUTINES::TIMING_SET_ON().

5.29.2.3 INTEGER,parameter KINDS::DPC = KIND((1.0_DP

Definition at line 74 of file kinds.f90.

5.29.2.4 INTEGER,parameter KINDS::SPC = KIND((1.0_SP

Definition at line 73 of file kinds.f90.

5.30 LISTS::DataType

Data type parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_INTG_TYPE = INTEGER_TYPE](#)
Integer data type for a list.
- INTEGER(INTG), parameter [LISTS::LIST_SP_TYPE = SINGLE_REAL_TYPE](#)
Single precision real data type for a list.
- INTEGER(INTG), parameter [LISTS::LIST_DP_TYPE = DOUBLE_REAL_TYPE](#)
Double precision real data type for a list.

5.30.1 Detailed Description

Data type parameters for a list.

See also:

[LISTS](#)

5.30.2 Variable Documentation

5.30.2.1 INTEGER(INTG),parameter [LISTS::LIST_DP_TYPE = DOUBLE_REAL_TYPE](#)

Double precision real data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 65 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_FINISH\(\)](#), [LISTS::LIST_DATA_TYPE_SET\(\)](#), [LISTS::LIST_DETACH_AND_DESTROY_DP\(\)](#), [LISTS::LIST_ITEM_ADD_DP1\(\)](#), [LISTS::LIST_ITEM_DELETE\(\)](#), [LISTS::LIST_ITEM_IN_LIST_DP1\(\)](#), and [LISTS::LIST_REMOVE_DUPLICATES\(\)](#).

5.30.2.2 INTEGER(INTG),parameter [LISTS::LIST_INTG_TYPE = INTEGER_TYPE](#)

Integer data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 63 of file lists.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#), [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE\(\)](#),

FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DETACH_AND_DESTROY_INTG(), LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_INTG1(), LISTS::LIST_REMOVE_DUPLICATES(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

5.30.2.3 INTEGER(INTG),parameter LISTS::LIST_SP_TYPE = SINGLE_REAL_TYPE

Single precision real data type for a list.

See also:

[LISTS::DataType](#),[LISTS](#)

Definition at line 64 of file lists.f90.

Referenced by LISTS::LIST_CREATE_FINISH(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DETACH_AND_DESTROY_SP(), LISTS::LIST_ITEM_ADD_SP1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_SP1(), and LISTS::LIST_REMOVE_DUPLICATES().

5.31 LISTS::SortingOrder

Sorting order parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_UNSORTED_TYPE = 1](#)
Unsorted list type.
- INTEGER(INTG), parameter [LISTS::LIST_SORT_ASCENDING_TYPE = 2](#)
Ascending order for sort.
- INTEGER(INTG), parameter [LISTS::LIST_SORT_DESCENDING_TYPE = 3](#)
Descending order for sort.

5.31.1 Detailed Description

Sorting order parameters for a list.

See also:

[LISTS](#)

5.31.2 Variable Documentation

5.31.2.1 INTEGER(INTG),parameter [LISTS::LIST_SORT_ASCENDING_TYPE = 2](#)

Ascending order for sort.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 73 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_START\(\)](#).

5.31.2.2 INTEGER(INTG),parameter [LISTS::LIST_SORT_DESCENDING_TYPE = 3](#)

Descending order for sort.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 74 of file lists.f90.

5.31.2.3 INTEGER(INTG),parameter LISTS::LIST_UNSORTED_TYPE = 1

Unsorted list type.

See also:

[LISTS::SortingOrder](#),[LISTS](#)

Definition at line 72 of file lists.f90.

5.32 LISTS::SortingMethod

Sorting method parameters for a list.

Variables

- INTEGER(INTG), parameter [LISTS::LIST_BUBBLE_SORT_METHOD = 1](#)
Bubble sort method.
- INTEGER(INTG), parameter [LISTS::LIST_SHELL_SORT_METHOD = 2](#)
Shell sort method.
- INTEGER(INTG), parameter [LISTS::LIST_HEAP_SORT_METHOD = 3](#)
Heap sort method.

5.32.1 Detailed Description

Sorting method parameters for a list.

See also:

[LISTS](#)

5.32.2 Variable Documentation

5.32.2.1 INTEGER(INTG),parameter [LISTS::LIST_BUBBLE_SORT_METHOD = 1](#)

Bubble sort method.

See also:

[LISTS_SortingMethod](#),[LISTS](#)

Definition at line 81 of file lists.f90.

5.32.2.2 INTEGER(INTG),parameter [LISTS::LIST_HEAP_SORT_METHOD = 3](#)

Heap sort method.

See also:

[LISTS_SortingMethod](#),[LISTS](#)

Definition at line 83 of file lists.f90.

Referenced by [LISTS::LIST_CREATE_START\(\)](#).

5.32.2.3 INTEGER(INTG),parameter LISTS::LIST_SHELL_SORT_METHOD = 2

Shell sort method.

See also:

LISTS_SortingMethod, [LISTS](#)

Definition at line 82 of file lists.f90.

5.33 MATRIX_VECTOR::DataTypes

Matrix vector data types.

Variables

- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE** = INTEGER_TYPE
Integer matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_SP_TYPE** = SINGLE_REAL_TYPE
Single precision real matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE** = DOUBLE_REAL_TYPE
Double precision real matrix-vector data type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_VECTOR_L_TYPE** = LOGICAL_TYPE
Logical matrix-vector data type.

5.33.1 Detailed Description

Matrix vector data types.

See also:

[MATRIX_VECTOR](#)

5.33.2 Variable Documentation

5.33.2.1 INTEGER(INTG),parameter **MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE** = DOUBLE_REAL_TYPE

Double precision real matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 151 of file matrix_vector.f90.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DATA_GET_DP(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP2(),

MATRIX_VECTOR::MATRIX_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP2(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_DP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), and MATRIX_VECTOR::VECTOR_VALUES_SET_DP1().

5.33.2.2 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE = INTEGER_TYPE

Integer matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 149 of file matrix_vector.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(), MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_INTG(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), and MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1().

5.33.2.3 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_L_TYPE = LOGICAL_TYPE

Logical matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 152 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_L(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L2(), MATRIX_VECTOR::MATRIX_VALUES_GET_L(), and MATRIX_VECTOR::MATRIX_VALUES_GET_L1().

MATRIX_VECTOR::MATRIX_VALUES_GET_L2(), MATRIX_VECTOR::MATRIX_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_VALUES_SET_L1(), MATRIX_VECTOR::MATRIX_VALUES_SET_L2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_L(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_L(), MATRIX_VECTOR::VECTOR_VALUES_GET_L1(), MATRIX_VECTOR::VECTOR_VALUES_SET_L(), and MATRIX_VECTOR::VECTOR_VALUES_SET_L1().

5.33.2.4 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_VECTOR_SP_TYPE = SINGLE_REAL_TYPE

Single precision real matrix-vector data type.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

Definition at line 150 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DATA_GET_SP(), MATRIX_VECTOR::MATRIX_DATA_TYPE_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP10(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP10(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP2(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), and MATRIX_VECTOR::VECTOR_VALUES_SET_SP1().

5.34 MATRIX_VECTOR::StorageTypes

Matrix-vector storage type parameters.

Variables

- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE = 0`
Matrix block storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_DIAGONAL_STORAGE_TYPE = 1`
Matrix diagonal storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE = 2`
Matrix column major storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_ROW_MAJOR_STORAGE_TYPE = 3`
Matrix row major storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE = 4`
Matrix compressed row storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE = 5`
Matrix compressed column storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_ROW_COLUMN_STORAGE_TYPE = 6`
Matrix row-column storage type.

5.34.1 Detailed Description

Matrix-vector storage type parameters.

See also:

`MATRIX_VECTOR_MatrixStorageStructures`, `MATRIX_VECTOR`

5.34.2 Variable Documentation

5.34.2.1 INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE = 0`

Matrix block storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 159 of file matrix_vector.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE().

5.34.2.2 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE = 2

Matrix column major storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 161 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

5.34.2.3 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE = 5

Matrix compressed column storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 164 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.34.2.4 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_COMPRESSED_ROW -
STORAGE_TYPE = 4**

Matrix compressed row storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 163 of file matrix_vector.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.34.2.5 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_DIAGONAL -
STORAGE_TYPE = 1**

Matrix diagonal storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 160 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.34.2.6 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_ROW_COLUMN -
STORAGE_TYPE = 6**

Matrix row-column storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

Definition at line 165 of file matrix_vector.f90.

Referenced by MATRIX_VECTOR::MATRIX_CREATE_FINISH(), MATRIX_VECTOR::MATRIX_DUPLICATE(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET().

**5.34.2.7 INTEGER(INTG),parameter MATRIX_VECTOR::MATRIX_ROW_MAJOR_-
STORAGE_TYPE = 3**

Matrix row major storage type.

See also:

[MATRIX_VECTOR::StorageTypes](#),[MATRIX_VECTOR](#)

Definition at line 162 of file matrix_vector.f90.

Referenced by `MATRIX_VECTOR::MATRIX_CREATE_FINISH()`, `MATRIX_VECTOR::MATRIX_-DUPLICATE()`, `MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET()`, `MATRIX_-VECTOR::MATRIX_OUTPUT()`, `MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND()`, `MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET()`, `MATRIX_VECTOR::MATRIX_-STORAGE_LOCATIONS_SET()`, `MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET()`.

5.35 MESH_ROUTINES::DecompositionTypes

The Decomposition types parameters.

Variables

- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_ALL_TYPE = 1](#)
The decomposition contains all elements.
- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_CALCULATED_TYPE = 2](#)
The element decomposition is calculated by graph partitioning.
- INTEGER(INTG), parameter [MESH_ROUTINES::DECOMPOSITION_USER_DEFINED_TYPE = 3](#)
The user will set the element decomposition.

5.35.1 Detailed Description

The Decomposition types parameters.

See also:

[MESH_ROUTINES](#)

5.35.2 Variable Documentation

5.35.2.1 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_ALL_TYPE = 1

The decomposition contains all elements.

See also:

[MESH_ROUTINES::DecompositionTypes](#),[MESH_ROUTINES](#)

Definition at line 73 of file mesh_routines.f90.

Referenced by [MESH_ROUTINES::DECOMPOSITION_CREATE_START\(\)](#), [MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE\(\)](#), and [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET\(\)](#).

5.35.2.2 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_CALCULATED_TYPE = 2

The element decomposition is calculated by graph partitioning.

See also:

[MESH_ROUTINES::DecompositionTypes](#),[MESH_ROUTINES](#)

Definition at line 74 of file mesh_routines.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), and MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

5.35.2.3 INTEGER(INTG),parameter MESH_ROUTINES::DECOMPOSITION_USER_DEFINED_TYPE = 3

The user will set the element decomposition.

See also:

[MESH_ROUTINES::DecompositionTypes](#), [MESH_ROUTINES](#)

Definition at line 75 of file mesh_routines.f90.

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), and MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

5.36 PROBLEM_CONSTANTS::SetupTypes

Setup type parameters.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE = 1**

Initial setup for a problem.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE = 2**

Solver setup for a problem.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP SOLUTION_TYPE = 3**

Solution parameters setup for a problem.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE = 4**

Solver setup for a problem.

5.36.1 Detailed Description

Setup type parameters.

See also:

[PROBLEM_CONSTANTS](#)

5.36.2 Variable Documentation

5.36.2.1 INTEGER(INTG),parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE = 2**

Solver setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_ROU](#)

Definition at line 107 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START().

**5.36.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_-
INITIAL_TYPE = 1**

Initial setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 106 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_CREATE_START().

**5.36.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_-
SOLUTION_TYPE = 3**

Solution parameters setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 108 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START().

**5.36.2.4 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_-
SOLVER_TYPE = 4**

Solver setup for a problem.

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 109 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START().

5.37 PROBLEM_CONSTANTS::SetupActionTypes

Setup action type parameters.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION = 1**
Start setup action.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION = 2**
Finish setup action.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION = 3**
Do setup action.

5.37.1 Detailed Description

Setup action type parameters.

See also:

[PROBLEM_CONSTANTS](#)

5.37.2 Variable Documentation

5.37.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION = 3

Do setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 118 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), and PROBLEM_ROUTINES::PROBLEM SOLUTION EQUATIONS_SET_ADD().

5.37.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION = 2

Finish setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 117 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH().

5.37.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION = 1

Start setup action.

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

Definition at line 116 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(), and PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START().

5.38 PROBLEM_CONSTANTS::SolutionOutputTypes

Solution output types.

Variables

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION NO_-
OUTPUT = 0
No output.
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION TIMING_-
OUTPUT = 1
Timing information output.
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION MATRIX_-
OUTPUT = 2
All below and solution matrices output.

5.38.1 Detailed Description

Solution output types.

See also:

[PROBLEM_CONSTANTS](#)

5.38.2 Variable Documentation

5.38.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_- MATRIX_OUTPUT = 2

All below and solution matrices output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 127 of file problem_constants.f90.

5.38.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_- NO_OUTPUT = 0

No output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#), [PROBLEM_CONSTANTS](#)

Definition at line 125 of file problem_constants.f90.

**5.38.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLUTION_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[PROBLEM_CONSTANTS::SolutionOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 126 of file problem_constants.f90.

5.39 PROBLEM_CONSTANTS::SolverOutputTypes

Solver output types.

Variables

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_NO_OUTPUT = 0**

No output.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_TIMING_OUTPUT = 1**

Timing information output.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_SOLVER_OUTPUT = 2**

All below and solver output.

5.39.1 Detailed Description

Solver output types.

See also:

[PROBLEM_CONSTANTS](#)

5.39.2 Variable Documentation

5.39.2.1 INTEGER(INTG),parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_NO_OUTPUT = 0**

No output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 135 of file problem_constants.f90.

5.39.2.2 INTEGER(INTG),parameter **PROBLEM_CONSTANTS::PROBLEM_SOLVER_SOLVER_OUTPUT = 2**

All below and solver output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 137 of file problem_constants.f90.

**5.39.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_-
TIMING_OUTPUT = 1**

Timing information output.

See also:

[PROBLEM_CONSTANTS::SolverOutputTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 136 of file problem_constants.f90.

5.40 PROBLEM_CONSTANTS::SolverSparsityTypes

Solution output types.

Variables

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_SPARSE_MATRICES = 1

Use sparse solver matrices.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_FULL_MATRICES = 2

Use fully populated solver matrices.

5.40.1 Detailed Description

Solution output types.

See also:

[PROBLEM_CONSTANTS](#)

5.40.2 Variable Documentation

5.40.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_FULL_MATRICES = 2

Use fully populated solver matrices.

See also:

[PROBLEM_CONSTANTS::SolverSparsityTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 145 of file problem_constants.f90.

5.40.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_SOLVER_SPARSE_MATRICES = 1

Use sparse solver matrices.

See also:

[PROBLEM_CONSTANTS::SolverSparsityTypes](#),[PROBLEM_CONSTANTS](#)

Definition at line 144 of file problem_constants.f90.

5.41 PROBLEM_ROUTINES::LinearityTypes

The linearity type parameters.

Variables

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_LINEARITIES = 3](#)

The number of problem linearity types defined.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_LINEAR = 1](#)

The problem is linear.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_NONLINEAR = 2](#)

The problem is non-linear.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_NONLINEAR_BCS = 3](#)

The problem has non-linear boundary conditions.

5.41.1 Detailed Description

The linearity type parameters.

See also:

[PROBLEM_ROUTINES](#)

5.41.2 Variable Documentation

5.41.2.1 INTEGER(INTG),parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_LINEARITIES = 3](#)

The number of problem linearity types defined.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 73 of file problem_routines.f90.

5.41.2.2 INTEGER(INTG),parameter [PROBLEM_ROUTINES::PROBLEM_LINEAR = 1](#)

The problem is linear.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 74 of file problem_routines.f90.

5.41.2.3 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_NONLINEAR = 2

The problem is non-linear.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 75 of file problem_routines.f90.

**5.41.2.4 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_NONLINEAR_-
BCS = 3**

The problem has non-linear boundary conditions.

See also:

[PROBLEM_ROUTINES::LinearityTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 76 of file problem_routines.f90.

5.42 PROBLEM_ROUTINES::TimeDepedenceTypes

The time dependence type parameters.

Variables

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_TIME_TYPES = 3](#)

The number of problem time dependence types defined.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_STATIC = 1](#)

The problem is static and has no time dependence.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_DYNAMIC = 2](#)

The problem is dynamic.

- INTEGER(INTG), parameter [PROBLEM_ROUTINES::PROBLEM_QUASISTATIC = 3](#)

The problem is quasi-static.

5.42.1 Detailed Description

The time dependence type parameters.

See also:

[PROBLEM_ROUTINES](#)

5.42.2 Variable Documentation

5.42.2.1 INTEGER(INTG),parameter [PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_TIME_TYPES = 3](#)

The number of problem time dependence types defined.

See also:

[PROBLEM_ROUTINES_TimeDependenceTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 83 of file problem_routines.f90.

5.42.2.2 INTEGER(INTG),parameter [PROBLEM_ROUTINES::PROBLEM_DYNAMIC = 2](#)

The problem is dynamic.

See also:

[PROBLEM_ROUTINES_TimeDependenceTypes](#),[PROBLEM_ROUTINES](#)

Definition at line 85 of file problem_routines.f90.

5.42.2.3 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_QUASISTATIC = 3

The problem is quasi-static.

See also:

PROBLEM_ROUTINES_TimeDependenceTypes,[PROBLEM_ROUTINES](#)

Definition at line 86 of file problem_routines.f90.

5.42.2.4 INTEGER(INTG),parameter PROBLEM_ROUTINES::PROBLEM_STATIC = 1

The problem is static and has no time dependence.

See also:

PROBLEM_ROUTINES_TimeDependenceTypes,[PROBLEM_ROUTINES](#)

Definition at line 84 of file problem_routines.f90.

5.43 SOLVER_ROUTINES::SolverTypes

The types of a problem solver.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_TYPE = 1](#)
A linear solution solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE = 2](#)
A nonlinear solution solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_TYPE = 3](#)
A time integration solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_TYPE = 4](#)
A eigenproblem type.

5.43.1 Detailed Description

The types of a problem solver.

See also:

[SOLVER_ROUTINES](#)

5.43.2 Variable Documentation

5.43.2.1 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_TYPE = 4

A eigenproblem type.

See also:

[SOLVER_ROUTINES::SolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 74 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_INITIALISE\(\)](#), [SOLVER_ROUTINES::SOLVER_LIBRARY_SET\(\)](#), and [SOLVER_ROUTINES::SOLVER_SOLVE\(\)](#).

5.43.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_LINEAR_TYPE = 1

Linear solution solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 71 of file solver_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.43.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE = 2

A nonlinear solution solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 72 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.43.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_TYPE = 3

A time integration solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 73 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.44 SOLVER_ROUTINES::SolverLibraries

The types of solver libraries.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_CMISS_LIBRARY = LIBRARY_CMISS_TYPE**
CMISS (internal) solver library.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY = LIBRARY_PETSC_TYPE**
PETSc solver library.

5.44.1 Detailed Description

The types of solver libraries.

See also:

[SOLVER_ROUTINES](#)

5.44.2 Variable Documentation

5.44.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_CMISS_LIBRARY = LIBRARY_CMISS_TYPE**

CMISS (internal) solver library.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 81 of file solver_routines.f90.

Referenced by **SOLVER_ROUTINES::SOLVER_LIBRARY_SET()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE()**, **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET()**, **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()**, and **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE()**.

5.44.2.2 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY = LIBRARY_PETSC_TYPE**

PETSc solver library.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 82 of file solver_routines.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(), SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE(), and SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE().

5.45 SOLVER_ROUTINES::LinearSolverTypes

The types of linear solvers.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE_TYPE = 1](#)

Direct linear solver type.

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE = 2](#)

Iterative linear solver type.

5.45.1 Detailed Description

The types of linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.45.2 Variable Documentation

5.45.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE_TYPE = 1](#)

Direct linear solver type.

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 89 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LIBRARY_SET\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET\(\)](#).

5.45.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE = 2](#)

Iterative linear solver type.

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 90 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_solve(), and SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET().

5.46 SOLVER_ROUTINES::DirectLinearSolverTypes

The types of direct linear solvers.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_LU = 1](#)
LU direct linear solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_CHOLESKY = 2](#)
Cholesky direct linear solver.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_DIRECT_SVD = 3](#)
SVD direct linear solver.

5.46.1 Detailed Description

The types of direct linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.46.2 Variable Documentation

5.46.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_DIRECT_CHOLESKY = 2](#)

Cholesky direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 98 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.46.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_DIRECT_LU = 1](#)

LU direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 97 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.46.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_DIRECT_SVD = 3

SVD direct linear solver.

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 99 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET\(\)](#).

5.47 SOLVER_ROUTINES::IterativeLinearSolverTypes

The types of iterative linear solvers.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_RICHARDSON = 1**
Richardson iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CHEBYCHEV = 2**
Chebychev iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJUGATE_GRADIENT = 3**
Conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICONJUGATE_GRADIENT = 4**
Bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_GMRES = 5**
Generalised minimum residual iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICGSTAB = 6**
Stabalised bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJGRAD_SQUARED = 7**
Conjugate gradient squared iterative solver type.

5.47.1 Detailed Description

The types of iterative linear solvers.

See also:

[SOLVER_ROUTINES](#)

5.47.2 Variable Documentation

5.47.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICGSTAB = 6**

Stabalised bi-conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 111 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- BICONJUGATE_GRADIENT = 4

Bi-conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 109 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CHEBYCHEV = 2

Chebychev iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 107 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CONJGRAD_SQUARED = 7

Conjugate gradient squared iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 112 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.5 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_- CONJUGATE_GRADIENT = 3

Conjugate gradient iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 108 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.6 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_GMRES = 5

Generalised minimum residual iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 110 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.47.2.7 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_RICHARDSON = 1

Richardson iterative solver type.

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

Definition at line 106 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET().

5.48 SOLVER_ROUTINES::IterativePreconditionerTypes

The types of iterative preconditioners.

Variables

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_NO_PRECONDITIONER = 0**
No preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_JACOBI_PRECONDITIONER = 1**
Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER = 2**
Iterative block Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_SOR_PRECONDITIONER = 3**
Successive over relaxation preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER = 4**
Incomplete Cholesky preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER = 5**
Incomplete LU preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER = 6**
Additive Schwarz preconditioner type.

5.48.1 Detailed Description

The types of iterative preconditioners.

See also:

[SOLVER_ROUTINES](#)

5.48.2 Variable Documentation

5.48.2.1 INTEGER(INTG),parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER = 6**

Additive Schwarz preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 125 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.48.2.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER = 2

Iterative block Jacobi preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 121 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.48.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER = 4

Incomplete Cholesky preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 123 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

5.48.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER = 5

Incomplete LU preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#),[SOLVER_ROUTINES](#)

Definition at line 124 of file solver_routines.f90.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET()`.

**5.48.2.5 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_-
JACOBI_PRECONDITIONER = 1**

Jacobi preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 120 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

**5.48.2.6 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_NO_-
PRECONDITIONER = 0**

No preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 119 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

**5.48.2.7 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ITERATIVE_SOR_-
PRECONDITIONER = 3**

Successive over relaxation preconditioner type.

See also:

[SOLVER_ROUTINES::IterativePreconditionerTypes](#), [SOLVER_ROUTINES](#)

Definition at line 122 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET\(\)](#).

5.49 SOLVER_ROUTINES::OutputTypes

The types of output.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_NO_OUTPUT = 0](#)
No output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_TIMING_OUTPUT = 1](#)
Timing output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_SOLVER_OUTPUT = 2](#)
Timing and solver specific output from the solver routines.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_MATRIX_OUTPUT = 3](#)
Timing and solver specific output and solution matrices output from the solver routines.

5.49.1 Detailed Description

The types of output.

See also:

[SOLVER_ROUTINES](#)

5.49.2 Variable Documentation

5.49.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_MATRIX_OUTPUT = 3](#)

Timing and solver specific output and solution matrices output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 135 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET\(\)](#), and [SOLVER_ROUTINES::SOLVER_SOLVE\(\)](#).

5.49.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_NO_OUTPUT = 0](#)

No output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 132 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_INITIALISE(), and SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET().

5.49.2.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_SOLVER_OUTPUT = 2

Timing and solver specific output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 134 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), and SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET().

5.49.2.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_TIMING_OUTPUT = 1

Timing output from the solver routines.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 133 of file solver_routines.f90.

Referenced by SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET(), and SOLVER_ROUTINES::SOLVER_SOLVE().

5.50 SOLVER_ROUTINES::SparsityTypes

The types of sparse solver matrices.

Variables

- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES = 1](#)
Use sparse solver matrices.
- INTEGER(INTG), parameter [SOLVER_ROUTINES::SOLVER_FULL_MATRICES = 2](#)
Use fully populated solver matrices.

5.50.1 Detailed Description

The types of sparse solver matrices.

See also:

[SOLVER_ROUTINES](#)

5.50.2 Variable Documentation

5.50.2.1 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_FULL_MATRICES = 2](#)

Use fully populated solver matrices.

See also:

[SOLVER_ROUTINES::SparsityTypes](#),[SOLVER_ROUTINES](#)

Definition at line 143 of file solver_routines.f90.

Referenced by [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET\(\)](#).

5.50.2.2 INTEGER(INTG),parameter [SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES = 1](#)

Use sparse solver matrices.

See also:

[SOLVER_ROUTINES::SparsityTypes](#),[SOLVER_ROUTINES](#)

Definition at line 142 of file solver_routines.f90.

Referenced by [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP\(\)](#), [SOLVER_ROUTINES::SOLVER_INITIALISE\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH\(\)](#), [SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH\(\)](#), and [SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET\(\)](#).

5.51 TREES::TreeNodeColourTypes

The colour of the tree nodes.

Variables

- INTEGER(INTG), parameter [TREES::TREE_BLACK_NODE = 0](#)
The black colour type for a tree node.

- INTEGER(INTG), parameter [TREES::TREE_RED_NODE = 1](#)
The red colour type for a tree node.

5.51.1 Detailed Description

The colour of the tree nodes.

See also:

[TREES](#)

5.51.2 Variable Documentation

5.51.2.1 INTEGER(INTG),parameter TREES::TREE_BLACK_NODE = 0

The black colour type for a tree node.

See also:

[TREES::TreeNodeColourTypes](#),[TREES](#)

Definition at line 62 of file trees.f90.

Referenced by [TREES::TREE_ITEM_DELETE\(\)](#), [TREES::TREE_ITEM_INSERT\(\)](#), and [TREES::TREE_NODE_INITIALISE\(\)](#).

5.51.2.2 INTEGER(INTG),parameter TREES::TREE_RED_NODE = 1

The red colour type for a tree node.

See also:

[TREES::TreeNodeColourTypes](#),[TREES](#)

Definition at line 63 of file trees.f90.

Referenced by [TREES::TREE_ITEM_DELETE\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

5.52 TREES::TreeNodeInsertStatus

The insert status for tree nodes.

Variables

- INTEGER(INTG), parameter [TREES::TREE_NODE_INSERT_SUCESSFUL = 1](#)
Successful insert status.
- INTEGER(INTG), parameter [TREES::TREE_NODE_DUPLICATE_KEY = 2](#)
Duplicate key found for those trees that do not allow duplicate keys.

5.52.1 Detailed Description

The insert status for tree nodes.

See also:

[TREES](#)

5.52.2 Variable Documentation

5.52.2.1 INTEGER(INTG),parameter [TREES::TREE_NODE_DUPLICATE_KEY = 2](#)

Duplicate key found for those trees that do not allow duplicate keys.

See also:

[TREES::TreeNodeInsertStatus](#), [TREES](#)

Definition at line 71 of file trees.f90.

Referenced by [TREES::TREE_ITEM_INSERT\(\)](#).

5.52.2.2 INTEGER(INTG),parameter [TREES::TREE_NODE_INSERT_SUCESSFUL = 1](#)

Successful insert status.

See also:

[TREES::TreeNodeInsertStatus](#), [TREES](#)

Definition at line 70 of file trees.f90.

Referenced by [NODE_ROUTINES::NODE_NUMBER_SET\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

5.53 TREES::TreeInsertTypes

The insert type for a tree.

Variables

- INTEGER(INTG), parameter [TREES::TREE_DUPLICATES_ALLOWED_TYPE = 1](#)
Duplicate keys allowed tree type.
- INTEGER(INTG), parameter [TREES::TREE_NO_DUPLICATES_ALLOWED = 2](#)
No duplicate keys allowed tree type.

5.53.1 Detailed Description

The insert type for a tree.

See also:

[TREES](#)

5.53.2 Variable Documentation

5.53.2.1 INTEGER(INTG),parameter [TREES::TREE_DUPLICATES_ALLOWED_TYPE = 1](#)

Duplicate keys allowed tree type.

See also:

[TREES::TreeInsertTypes](#),[TREES](#)

Definition at line 78 of file trees.f90.

Referenced by [TREES::TREE_CREATE_START\(\)](#), and [TREES::TREE_INSERT_TYPE_SET\(\)](#).

5.53.2.2 INTEGER(INTG),parameter [TREES::TREE_NO_DUPLICATES_ALLOWED = 2](#)

No duplicate keys allowed tree type.

See also:

[TREES::TreeInsertTypes](#),[TREES](#)

Definition at line 79 of file trees.f90.

Referenced by [MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE\(\)](#), [NODE_ROUTINES::NODES_CREATE_START\(\)](#), [TREES::TREE_INSERT_TYPE_SET\(\)](#), and [TREES::TREE_ITEM_INSERT\(\)](#).

Chapter 6

Namespace Documentation

6.1 BASE_ROUTINES Namespace Reference

This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

Classes

- struct [ROUTINE_LIST_ITEM_TYPE](#)
Contains information for an item in the routine list for diagnostics or timing.
- struct [ROUTINE_LIST_TYPE](#)
Contains information for the routine list for diagnostics or timing.
- struct [ROUTINE_STACK_ITEM_TYPE](#)
Contains information for an item in the routine invocation stack.
- struct [ROUTINE_STACK_TYPE](#)
Contains information for the routine invocation stack.
- interface [interface](#)
- interface [FLAG_ERROR](#)
Flags an error condition.
- interface [FLAG_WARNING](#)
Flags a warning to the user.

Functions

- subroutine [ENTERS](#) (NAME, ERR, ERROR,*)
Records the entry into the named procedure and initialises the error code.
- subroutine [ERRORS](#) (NAME, ERR, ERROR)

Records the exiting error of the subroutine.

- subroutine **EXITS** (NAME)

Records the exit out of the named procedure.

- subroutine **FLAG_ERROR_C** (STRING, ERR, ERROR,*)

Sets the error string specified by a character string and flags an error.

- subroutine **FLAG_ERROR_VS** (STRING, ERR, ERROR,*)

Sets the error string specified by a varying string and flags an error.

- subroutine **FLAG_WARNING_C** (STRING, ERR, ERROR,*)

Writes a warning message specified by a character string to the user.

- subroutine **FLAG_WARNING_VS** (STRING, ERR, ERROR,*)

Writes a warning message specified by a varying string to the user.

- subroutine **BASE_ROUTINES_FINALISE** (ERR, ERROR,*)

Finalises the base_routines module and deallocates all memory.

- subroutine **BASE_ROUTINES_INITIALISE** (ERR, ERROR,*)

Initialises the variables required for the base_routines module.

- subroutine **DIAGNOSTICS_SET_OFF** (ERR, ERROR,*)

Sets diagnostics off.

- subroutine **DIAGNOSTICS_SET_ON** (DIAG_TYPE, LEVEL_LIST, DIAG_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets diagnostics on.

- subroutine **OUTPUT_SET_OFF** (ERR, ERROR,*)

Sets writes file echo output off.

- subroutine **OUTPUT_SET_ON** (ECHO_FILENAME, ERR, ERROR,*)

Sets writes file echo output on.

- subroutine **TIMING_SET_OFF** (ERR, ERROR,*)

Sets timing off.

- subroutine **TIMING_SET_ON** (TIMING_TYPE, TIMING_SUMMARY_FLAG, TIMING_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets timing on.

- subroutine **TIMING_SUMMARY_OUTPUT** (ERR, ERROR,*)

Outputs the timing summary.

- subroutine **WRITE_STR** (ID, ERR, ERROR,*)

Writes the output string to a specified output stream.

Variables

- INTEGER(INTG), parameter **MAX_OUTPUT_LINES** = 500
Maximum number of lines that can be output.
- INTEGER(INTG), parameter **GENERAL_OUTPUT_TYPE** = 1
General output type.
- INTEGER(INTG), parameter **DIAGNOSTIC_OUTPUT_TYPE** = 2
Diagnostic output type.
- INTEGER(INTG), parameter **TIMING_OUTPUT_TYPE** = 3
Timing output type.
- INTEGER(INTG), parameter **ERROR_OUTPUT_TYPE** = 4
Error output type.
- INTEGER(INTG), parameter **HELP_OUTPUT_TYPE** = 5
Help output type.
- INTEGER(INTG), parameter **ECHO_FILE_UNIT** = 10
File unit for echo files.
- INTEGER(INTG), parameter **DIAGNOSTICS_FILE_UNIT** = 11
File unit for diagnostic files.
- INTEGER(INTG), parameter **TIMING_FILE_UNIT** = 12
File unit for timing files.
- INTEGER(INTG), parameter **LEARN_FILE_UNIT** = 13
File unit for learn files.
- INTEGER(INTG), parameter **IO1_FILE_UNIT** = 21
File unit for general IO 1 files.
- INTEGER(INTG), parameter **IO2_FILE_UNIT** = 22
File unit for general IO 2 files.
- INTEGER(INTG), parameter **IO3_FILE_UNIT** = 23
File unit for general IO 3 files.
- INTEGER(INTG), parameter **IO4_FILE_UNIT** = 24
File unit for general IO 4 files.
- INTEGER(INTG), parameter **IO5_FILE_UNIT** = 25
File unit for general IO 5 files.
- INTEGER(INTG), parameter **TEMPORARY_FILE_UNIT** = 80
File unit for temporary files.

- INTEGER(INTG), parameter **OPEN_COMFILE_UNIT** = 90
File unit for open command files.
- INTEGER(INTG), parameter **START_READ_COMFILE_UNIT** = 90
First file unit for read command files.
- INTEGER(INTG), parameter **STOP_READ_COMFILE_UNIT** = 99
Last file unit for read command files.
- INTEGER(INTG), parameter **ALL_DIAG_TYPE** = 1
Type for setting diagnostic output in all routines.
- INTEGER(INTG), parameter **IN_DIAG_TYPE** = 2
Type for setting diagnostic output in one routine.
- INTEGER(INTG), parameter **FROM_DIAG_TYPE** = 3
Type for setting diagnostic output from one routine downwards.
- INTEGER(INTG), parameter **ALL_TIMING_TYPE** = 1
Type for setting timing output in all routines.
- INTEGER(INTG), parameter **IN_TIMING_TYPE** = 2
Type for setting timing output in one routine.
- INTEGER(INTG), parameter **FROM_TIMING_TYPE** = 3
Type for setting timing output from one routine downwards.
- LOGICAL, save **DIAGNOSTICS**
.TRUE. if diagnostic output is required in any routines.
- LOGICAL, save **DIAGNOSTICS1**
.TRUE. if level 1 diagnostic output is active in the current routine
- LOGICAL, save **DIAGNOSTICS2**
.TRUE. if level 2 diagnostic output is active in the current routine
- LOGICAL, save **DIAGNOSTICS3**
.TRUE. if level 3 diagnostic output is active in the current routine
- LOGICAL, save **DIAGNOSTICS4**
.TRUE. if level 4 diagnostic output is active in the current routine
- LOGICAL, save **DIAGNOSTICS5**
.TRUE. if level 5 diagnostic output is active in the current routine
- LOGICAL, save **DIAGNOSTICS_LEVEL1**
.TRUE. if the user has requested level 1 diagnostic output to be active
- LOGICAL, save **DIAGNOSTICS_LEVEL2**
.TRUE. if the user has requested level 2 diagnostic output to be active

- LOGICAL, save [DIAGNOSTICS_LEVEL3](#)
.TRUE. if the user has requested level 3 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL4](#)
.TRUE. if the user has requested level 4 diagnostic output to be active
- LOGICAL, save [DIAGNOSTICS_LEVEL5](#)
.TRUE. if the user has requested level 5 diagnostic output to be active
- LOGICAL, save [DIAG_ALL_SUBROUTINES](#)
.TRUE. if diagnostic output is required in all routines
- LOGICAL, save [DIAG_FROM_SUBROUTINE](#)
.TRUE. if diagnostic output is required from a particular routine
- LOGICAL, save [DIAG_FILE_OPEN](#)
.TRUE. if the diagnostic output file is open
- LOGICAL, save [DIAG_OR_TIMING](#)
.TRUE. if diagnostics or time is .TRUE.
- LOGICAL, save [ECHO_OUTPUT](#)
.TRUE. if all output is to be echoed to the echo file
- LOGICAL, save [TIMING](#)
.TRUE. if timing output is required in any routines.
- LOGICAL, save [TIMING_SUMMARY](#)
.TRUE. if timing output will be summary form via a TIMING_SUMMARY_OUTPUT call otherwise timing will be output for routines when the routine exits
- LOGICAL, save [TIMING_ALL_SUBROUTINES](#)
.TRUE. if timing output is required in all routines
- LOGICAL, save [TIMING_FROM_SUBROUTINE](#)
.TRUE. if timing output is required from a particular routine
- LOGICAL, save [TIMING_FILE_OPEN](#)
.TRUE. if the timing output file is open
- CHARACTER(LEN=MAXSTRLEN), save [OP_STRING](#)
The array of lines to output.
- TYPE([ROUTINE_LIST_TYPE](#)), save [DIAG_ROUTINE_LIST](#)
The list of routines for which diagnostic output is required.
- TYPE([ROUTINE_LIST_TYPE](#)), save [TIMING_ROUTINE_LIST](#)
The list of routines for which timing output is required.

- TYPE(**ROUTINE_STACK_TYPE**), save **ROUTINE_STACK**

The routine invocation stack.

6.1.1 Detailed Description

This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

6.1.2 Function Documentation

6.1.2.1 subroutine **BASE_ROUTINES::BASE_ROUTINES_FINALISE** (**INTEGER(INTG,intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**)

Finalises the base_routines module and deallocates all memory.

Todo

Finish this routine and deallocate memory.

Parameters:

ERR The error code

ERROR The error string

Definition at line 552 of file base_routines.f90.

Referenced by CMISS::CMISS_FINALISE().

Here is the caller graph for this function:

6.1.2.2 subroutine **BASE_ROUTINES::BASE_ROUTINES_INITIALISE** (**INTEGER(INTG,intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**)

Initialises the variables required for the base_routines module.

Parameters:

ERR The error code

ERROR The error string

Definition at line 571 of file base_routines.f90.

References KINDS::_DP, KINDS::_SP, and MACHINE_CONSTANTS::MACHINE_OS.

Referenced by CMISS::CMISS_INITIALISE().

Here is the caller graph for this function:

6.1.2.3 subroutine **BASE_ROUTINES::DIAGNOSTICS_SET_OFF** (**INTEGER(INTG,intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**)

Sets diagnostics off.

See also:

[BASE_ROUTINES::DIAGNOSTICS_SET_ON](#)

Parameters:

ERR The error code

ERROR The error string

Definition at line 640 of file base_routines.f90.

6.1.2.4 subroutine BASE_ROUTINES::DIAGNOSTICS_SET_ON
(INTEGER(INTG),intent(in) DIAG_TYPE, INTEGER(INTG),dimension(:),intent(in) LEVEL_LIST, CHARACTER(LEN=),intent(in) DIAG_FILENAME, CHARACTER(LEN=*),dimension(:),intent(in) ROUTINE_LIST, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]*

Sets diagnostics on.

See also:

[BASE_ROUTINES::DIAGNOSTICS_SET_OFF](#)

Parameters:

DIAG_TYPE The type of diagnostics to set on

See also:

[BASE_ROUTINES::DiagnosticTypes](#)

LEVEL_LIST The list of diagnostic levels to set on

DIAG_FILENAME If present the name of the file to output diagnostic information to. If omitted the diagnostic output is sent to the screen

ROUTINE_LIST The list of routines to set diagnostics on in.

ERR The error code

ERROR The error string

Definition at line 695 of file base_routines.f90.

6.1.2.5 subroutine BASE_ROUTINES::ENTERS (CHARACTER(LEN=*),intent(in) NAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Records the entry into the named procedure and initialises the error code.

See also:

[BASE_ROUTINES::EXITS](#)

Parameters:

NAME The name of the routine being entered

ERR The error code

ERROR The error string

Definition at line 218 of file base_routines.f90.

References KINDS::_SP.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_-CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_-SETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_-TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARY_-FILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONAL_-ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_-ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_-NODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_-TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_-TYPE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_-NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_-FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_-RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_-CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_-START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_-SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_-TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE(), TIMER::CPU_TIMER(), MATHS::CROSS_PRODUCT_DP(), MATHS::CROSS_PRODUCT_INTG(), MATHS::CROSS_PRODUCT_SP(), COORDINATE_ROUTINES::D2ZX_DP(), MATHS::D_CROSS_PRODUCT_DP(), MATHS::D_CROSS_PRODUCT_-INTG(), MATHS::D_CROSS_PRODUCT_SP(), MESH_ROUTINES::DECOMPOSITION_-CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_ROUTINES::DECOMPOSITION_DESTROY(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_-DOMAIN_SET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_-

```

NUMBER_SET(),           MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_-
SET(),     MATHS::DETERMINANT_FULL_DP(),    MATHS::DETERMINANT_FULL_INTG(),
MATHS::DETERMINANT_FULL_SP(),          DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_FINALISE(),          DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_INITIALISE(),        DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
LOCAL_FROM_GLOBAL_CALCULATE(),      DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
MAPPING_FINALISE(),                DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_FINALISE(),                 DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_INITIALISE(),               DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
INITIALISE(),         MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(),   MESH_-
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(),   MESH_ROUTINES::DOMAIN_-
TOPOLOGY_CALCULATE(),             MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_ELEMENT_FINALISE(),       MESH_ROUTINES::DOMAIN_TOPOLOGY_-_
ELEMENT_INITIALISE(),             MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-_
FINALISE(),           MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(),      MESH_ROUTINES::DOMAIN_-
TOPOLOGY_INITIALISE(),            MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_-_
FROM_MESH(),       MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(),   MESH_-
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(),     MESH_ROUTINES::DOMAIN_-
TOPOLOGY_LINES_FINALISE(),         MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_-_
INITIALISE(),         MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(),   MESH_-
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(),     MESH_ROUTINES::DOMAIN_-
TOPOLOGY_NODES_FINALISE(),        MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_-_
INITIALISE(),         MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_-_
ELEMENTS_CALCULATE(),           COORDINATE_ROUTINES::DXZ_DP(),      COORDINATE_-
ROUTINES::DZX_DP(),             MATHS::EIGENVALUE_FULL_DP(),    MATHS::EIGENVALUE_-_
FULL_SP(),       MATHS::EIGENVECTOR_FULL_DP(),    MATHS::EIGENVECTOR_FULL_-_
SP(),     EQUATIONS_SET_ROUTINES::EQ(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
INTERPOLATION_FINALISE(),        EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_-_
INITIALISE(),         EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-_
FINISH(),           EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-_
START(),            EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_FINALISE(),   EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_ANALYTIC_INITIALISE(),      EQUATIONS_SET_-_
ROUTINES::EQUATIONS_SET_ASSEMBLE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_ASSEMBLE_LINEAR_STATIC_FEM(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
BACKSUBSTITUTE(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(),      EQUATIONS_SET_-_
ROUTINES::EQUATIONS_SET_DEPENDENT_(),   EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_FINISH(),       EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_START(),       EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
DEPENDENT_DEPENDENT_FIELD_GET(),     EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_DESTROY(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
DEPENDENT_FINALISE(),            EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_-_
INITIALISE(),         EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_SCALING_-_
SET(),     EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DESTROY(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_FINALISE(),        EQUATIONS_SET_-_
ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE(),      EQUATIONS_SET_-_
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE(), EQUATIONS_SET_-_
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE(), EQUATIONS_SET_-_
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE(),

```

```

EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_-
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_-
TYPE_SET(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_SPARSITY_-
TYPE_SET(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_-
DATA_FINALISE(),   EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_-
TIME_DATA_INITIALISE(),   EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF5(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_GEOMETRY_FINALISE(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_GEOMETRY_INITIALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_INITIALISE(),    EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_MATERIALS_COMPONENT_INTERPOLATION_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_-_
SET(),            EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-_
FINISH(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-_
START(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_FINALISE(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_MATERIALS_INITIALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_MATERIALS_SCALING_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SETUP(),    EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_SOURCE_CREATE_FINISH(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SOURCE_CREATE_START(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_-
DESTROY(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_INITIALISE(),    EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_SOURCE_SCALING_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SPECIFICAT(),    EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_USER_NUMBER_FIND(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE(),    F90C::F2CSTRING(),
FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(),    FINITE_-
ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_INITIALISE(),    FIELD_ROUTINES::FI(),
FIELD_IO_ROUTINES::FIE(),    FIELD_ROUTINES::FIELD_COMPONENT_INTER(),    FIELD_-
ROUTINES::FIELD_COMPONENT_MESH_COM(),    FIELD_ROUTINES::FIELD_CREATE_-_
FINISH(),          FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(),    FIELD_-
ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(),    FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_SET_NUMBER(),    FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_-_
PTR(),    FIELD_ROUTINES::FIELD_DESTROY(),    FIELD_ROUTINES::FIELD_DIMENSION_SET_-_
NUMBER(),    FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR(),    FIELD_ROUTINES::FIELD_-_
INTERPOLATE_GAUSS(),    FIELD_ROUTINES::FIELD_INTERPOLATE_XI(),    FIELD_-_
ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(),    FIELD_ROUTINES::FIELD_-_
INTERPOLATED_POINT_INITIALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-_
METRICS_CALCULATE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_-_
FINALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),    FIELD_-_
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(),    FIELD_ROUTINES::FIELD_-_
INTERPOLATION_PARAMETERS_INITIALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATION_-_
PARAMETERS_LINE_GET(),    FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_-

```

LABEL(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_INFO(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(), FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING(), FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_DERIVATIVE_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE(), FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT(), FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE(), FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE(), FIELD_ROUTINES::FIELD_VARIABLES_FINALISE(), FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE(), FIELD_ROUTINES::FIELDS_FINALISE(), FIELD_ROUTINES::FIELDS_INITIALISE(), GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_FINISH(), GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_START(), GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY(), GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(), GENERATED_MESH_ROUTINES::GENERATED_MESH_INITALISE(), GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET(), SORTING::HEAP_SORT_DP(), SORTING::HEAP_SORT_INTG(), SORTING::HEAP_SORT_SP(), BINARY_FILE::INQUIRE_EOF_BINARY_FILE(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_DETACH_AND_DESTROY_DP(), LISTS::LIST_DETACH_AND_DESTROY_INTG(), LISTS::LIST_DETACH_AND_DESTROY_SP(), LISTS::LIST_FINALISE(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INITIALISE(), LISTS::LIST_ITEM_ADD_DP1(), LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_ITEM_ADD_SP1(), LISTS::LIST_ITEM_DELETE(), LISTS::LIST_ITEM_IN_LIST_DP1(), LISTS::LIST_ITEM_IN_LIST_INTG1(), LISTS::LIST_ITEM_IN_LIST_SP1(), LISTS::LIST_NUMBER_OF_ITEMS_GET(), LISTS::LIST_REMOVE_DUPLICATES(), LISTS::LIST_SEARCH_DP_ARRAY(), LISTS::LIST_SEARCH_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_DP_ARRAY(), LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_

SP_ARRAY(), LISTS::LIST_SEARCH_SP_ARRAY(), LISTS::LIST_SORT_BUBBLE_DP_ARRAY(),
 LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(), LISTS::LIST_SORT_BUBBLE_SP_ARRAY(),
 LISTS::LIST_SORT_DP_ARRAY(), LISTS::LIST_SORT_HEAP_DP_ARRAY(), LISTS::LIST_
 SORT_HEAP_INTG_ARRAY(), LISTS::LIST_SORT_HEAP_SP_ARRAY(), LISTS::LIST_
 SORT_INTG_ARRAY(), LISTS::LIST_SORT_SHELL_DP_ARRAY(), LISTS::LIST_SORT_
 SHELL_INTG_ARRAY(), LISTS::LIST_SORT_SHELL_SP_ARRAY(), LISTS::LIST_SORT_
 SP_ARRAY(), STRINGS::LIST_TO_CHARACTER_C(), STRINGS::LIST_TO_CHARACTER_
 DP(), STRINGS::LIST_TO_CHARACTER_INTG(), STRINGS::LIST_TO_CHARACTER_
 L(), STRINGS::LIST_TO_CHARACTER_LINTG(), STRINGS::LIST_TO_CHARACTER_
 SP(), STRINGS::LOGICAL_TO_CHARACTER(), STRINGS::LOGICAL_TO_VSTRING(),
 MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_ALL_
 VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(), MATRIX_
 VECTOR::MATRIX_ALL_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_CREATE_FINISH(),
 MATRIX_VECTOR::MATRIX_CREATE_START(), MATRIX_VECTOR::MATRIX_DATA_GET_
 DP(), MATRIX_VECTOR::MATRIX_DATA_GET_INTG(), MATRIX_VECTOR::MATRIX_DATA_
 GET_L(), MATRIX_VECTOR::MATRIX_DATA_GET_SP(), MATRIX_VECTOR::MATRIX_DATA_
 TYPE_SET(), MATRIX_VECTOR::MATRIX_DESTROY(), MATRIX_VECTOR::MATRIX_
 DUPLICATE(), MATRIX_VECTOR::MATRIX_FINALISE(), MATRIX_VECTOR::MATRIX_
 INITIALISE(), MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(), MATRIX_
 VECTOR::MATRIX_MAX_SIZE_SET(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_
 SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_SIZE_SET(),
 MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_
 STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_
 SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_GET(), MATRIX_VECTOR::MATRIX_
 STORAGE_TYPE_SET(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(), MATRIX_
 VECTOR::MATRIX_VALUES_ADD_DP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_
 DP2(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG(), MATRIX_VECTOR::MATRIX_
 VALUES_ADD_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(), MATRIX_
 VECTOR::MATRIX_VALUES_ADD_L(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(),
 MATRIX_VECTOR::MATRIX_VALUES_ADD_L2(), MATRIX_VECTOR::MATRIX_VALUES_
 ADD_SP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(), MATRIX_VECTOR::MATRIX_
 VALUES_ADD_SP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_
 VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_
 DP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(), MATRIX_VECTOR::MATRIX_
 VALUES_GET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(), MATRIX_
 VECTOR::MATRIX_VALUES_GET_L(), MATRIX_VECTOR::MATRIX_VALUES_GET_L1(),
 MATRIX_VECTOR::MATRIX_VALUES_GET_L2(), MATRIX_VECTOR::MATRIX_VALUES_
 GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(), MATRIX_VECTOR::MATRIX_
 VALUES_GET_SP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP(), MATRIX_
 VECTOR::MATRIX_VALUES_SET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_SET_
 DP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_
 VALUES_SET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(), MATRIX_
 VECTOR::MATRIX_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_VALUES_SET_
 L1(), MATRIX_VECTOR::MATRIX_VALUES_SET_L2(), MATRIX_VECTOR::MATRIX_
 VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(), MATRIX_
 VECTOR::MATRIX_VALUES_SET_SP2(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_
 ROUTINES::MESH_CREATE_REGULAR(), MESH_ROUTINES::MESH_CREATE_START(),
 MESH_ROUTINES::MESH_DESTROY(), MESH_ROUTINES::MESH_FINALISE(), MESH_
 ROUTINES::MESH_INITIALISE(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_
 SET_NUMBER(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(), MESH_
 ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(), MESH_ROUTINES::MESH_
 NUMBER_OF_ELEMENTS_SET_PTR(), MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(),
 MESH_ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(), MESH_ROUTINES::MESH_
 TOPOLOGY_DOFS_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(),

```

MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(), MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_USER_NUMBER_FIND(), MESH_ROUTINES::MESHS_FINALISE(), MESH_ROUTINES::MESHS_INITIALISE(), CMISS_MPI::MPI_ERROR_CHECK(), NODE_ROUTINES::NODE_CHECK_EXISTS(), NODE_ROUTINES::NODE_DESTROY(), NODE_ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_ROUTINES::NODE_NUMBER_SET(), NODE_ROUTINES::NODES_CREATE_FINISH(), NODE_ROUTINES::NODES_CREATE_START(), NODE_ROUTINES::NODES_FINALISE(), NODE_ROUTINES::NODES_INITIALISE(), MATHS::NORMALISE_DP(), MATHS::NORMALISE_SP(), STRINGS::NUMBER_TO_CHARACTER_DP(), STRINGS::NUMBER_TO_CHARACTER_INTG(), STRINGS::NUMBER_TO_CHARACTER_LINTG(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(), BINARY_FILE::OPEN_BINARY_FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_PARMETIS::PARMETIS_PARTKWAY(), CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON(), CMISS_PETSC::PETSC_FINALIZE(), CMISS_PETSC::PETSC_INITIALIZE(), CMISS_PETSC::PETSC_ISDESTROY(), CMISS_PETSC::PETSC_ISFINALISE(), CMISS_PETSC::PETSC_ISINITIALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGDESTROY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGFINALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE(), CMISS_PETSC::PETSC_KSPCREATE(), CMISS_PETSC::PETSC_KSPDESTROY(), CMISS_PETSC::PETSC_KSPFINALISE(), CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(), CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_PETSC::PETSC_KSPGETRESIDUALNORM(), CMISS_PETSC::PETSC_KSPINITIALISE(), CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(), CMISS_PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PETSC_KSPSETUP(), CMISS_PETSC::PETSC_KSPSOLVE(), CMISS_PETSC::PETSC_LOGPRINTSUMMARY(), CMISS_PETSC::PETSC_MATASSEMBLYBEGIN(), CMISS_PETSC::PETSC_MATASSEMBLYEND(), CMISS_PETSC::PETSC_MATCREATE(), CMISS_PETSC::PETSC_MATCREATEMPIAIJ(), CMISS_PETSC::PETSC_MATCREATETEMPIDENSE(), CMISS_PETSC::PETSC_MATCREATESEQAIJ(), CMISS_PETSC::PETSC_MATCREATESEQDENSE(), CMISS_PETSC::PETSC_MATDESTROY(), CMISS_PETSC::PETSC_MATFINALISE(), CMISS_PETSC::PETSC_MATGETARRAY(), CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_MATRESTOREARRAY(), CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(), CMISS_PETSC::PETSC_MATSETOPTION(), CMISS_PETSC::PETSC_MATSETSIZES(), CMISS_

```

PETSC::PETSC_MATSETVALUE(), CMISS_PETSC::PETSC_MATSETVALUELOCAL(), CMISS_PETSC::PETSC_MATSETVALUES(), CMISS_PETSC::PETSC_MATSETVALUESLOCAL(),
 CMISS_PETSC::PETSC_MATVIEW(), CMISS_PETSC::PETSC_MATZEROENTRIES(),
 CMISS_PETSC::PETSC_PCFINALISE(), CMISS_PETSC::PETSC_PCINITIALISE(), CMISS_PETSC::PETSC_PCSETTYPE(), CMISS_PETSC::PETSC_VECASSEMBLYBEGIN(),
 CMISS_PETSC::PETSC_VECASSEMBLYEND(), CMISS_PETSC::PETSC_VECCREATE(), CMISS_PETSC::PETSC_VECCREATEGHOST(), CMISS_PETSC::PETSC_VECCREATEGHOSTWITHARRAY(),
 CMISS_PETSC::PETSC_VECCREATEMPI(), CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY(), CMISS_PETSC::PETSC_VECCREATESEQ(),
 CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY(), CMISS_PETSC::PETSC_VECDESTROY(), CMISS_PETSC::PETSC_VECDUPLICATE(),
 CMISS_PETSC::PETSC_VECFINALISE(), CMISS_PETSC::PETSC_VECGETARRAY(), CMISS_PETSC::PETSC_VECGETARRAYF90(),
 CMISS_PETSC::PETSC_VECGETLOCALSIZE(), CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_VECGETSIZE(),
 CMISS_PETSC::PETSC_VECGETVALUES(), CMISS_PETSC::PETSC_VECHOSTGETLOCALFORM(), CMISS_PETSC::PETSC_VECGHOSTRESTORELOCALFORM(),
 CMISS_PETSC::PETSC_VECGHOSTUPDATEBEGIN(), CMISS_PETSC::PETSC_VECGHOSTUPDATEEND(), CMISS_PETSC::PETSC_VECINITIALISE(),
 CMISS_PETSC::PETSC_VCRESTOREARRAY(), CMISS_PETSC::PETSC_VCRESTOREARRAYF90(), CMISS_PETSC::PETSC_VECSET(),
 CMISS_PETSC::PETSC_VECSETFROMOPTIONS(), CMISS_PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING(), CMISS_PETSC::PETSC_VECSETSIZES(),
 CMISS_PETSC::PETSC_VECSETVALUES(), CMISS_PETSC::PETSC_VECSETVALUESLOCAL(), CMISS_PETSC::PETSC_VECVIEW(),
 PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_CONTROL_DESTROY(), PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE(),
 PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER(), PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR(),
 PROBLEM_ROUTINES::PROBLEM_FINALISE(), PROBLEM_ROUTINES::PROBLEM_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SETUP(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_FINALISE(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTION_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_solve(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(),
 PROBLEM_ROUTINES::PROBLEM_SOLVE(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(),
 PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(), PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(),
 PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR(),
 PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND(), PROBLEMS_FINALISE(), PROBLEM_ROUTINES::PROBLEMS_INITIALISE(),
 BINARY_FILE::READ_BINARY_FILE_CHARACTER(), BINARY_FILE::READ_BINARY_FILE_DP(), BINARY_FILE::READ_BINARY_FILE_DP1(),
 BINARY_FILE::READ_BINARY_FILE_DPC(), BINARY_FILE::READ_BINARY_FILE_DPC1(), BINARY_FILE::READ_BINARY_FILE_INTG(),
 BINARY_FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_FILE::READ_BINARY_FILE_LINTG1(),
 BINARY_FILE::READ_BINARY_FILE_LOGICAL(), BINARY_FILE::READ_BINARY_FILE_LOGICAL1(), BINARY_FILE::READ_BINARY_FILE_SINTG(),
 BINARY_FILE::READ_BINARY_FILE_SINTG1(), BINARY_FILE::READ_BINARY_FILE_SP(), BINARY_FILE::READ_BINARY_FILE_SP1(),
 BINARY_FILE::READ_BINARY_FILE_SPC(), BINARY_FILE::READ_BINARY_FILE_SPC1(),
 BINARY_FILE::READ_BINARY_TAG_HEADER(), REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET(),
 REGION_ROUTINES::REGION_COORDINATE_SYSTEM_

```
SET_NUMBER(),           REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR(),
REGION_ROUTINES::REGION_CREATE_FINISH(),   REGION_ROUTINES::REGION_CREATE_
-START(),   REGION_ROUTINES::REGION_DESTROY(),   REGION_ROUTINES::REGION_
-LABEL_GET(),   REGION_ROUTINES::REGION_LABEL_SET_NUMBER(),   REGION-
ROUTINES::REGION_LABEL_SET_PTR(),   REGION_ROUTINES::REGION_SUB_REGION_
-CREATE_FINISH(),   REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(),
REGION_ROUTINES::REGION_USER_NUMBER_FIND(),   REGION_ROUTINES::REGION-
USER_NUMBER_FIND_PTR(),   REGION_ROUTINES::REGIONS_FINALISE(),   REGION-
ROUTINES::REGIONS_INITIALISE(),   BINARY_FILE::RESET_BINARY_NUMBER_TAGS(),
BINARY_FILE::SET_BINARY_FILE(),   SORTING::SHELL_SORT_DP(),   SORTING::SHELL-
SORT_INTG(),   SORTING::SHELL_SORT_SP(),   BINARY_FILE::SKIP_BINARY_FILE(),   BINARY-
FILE::SKIP_BINARY_TAGS(),   BINARY_FILE::SKIP_CM_BINARY_HEADER(),   MATHS::SOLVE-
SMALL_LINEAR_SYSTEM_DP(),   MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(),   SOLVER-
ROUTINES::SOLVER_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_CREATE-
START(),   SOLVER_ROUTINES::SOLVER_DESTROY(),   SOLVER_ROUTINES::SOLVER-
EIGENPROBLEM_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_EIGENPROBLEM-
FINALISE(),   SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(),   SOLVER-
ROUTINES::SOLVER_EIGENPROBLEM_SOLVE(),   SOLVER_ROUTINES::SOLVER_FINALISE(),
SOLVER_ROUTINES::SOLVER_INITIALISE(),   SOLVER_ROUTINES::SOLVER_LIBRARY_SET(),
SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER-
LINEAR_DIRECT_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT-
FINALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(),   SOLVER-
ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(),   SOLVER_ROUTINES::SOLVER_LINEAR-
DIRECT_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(),   SOLVER-
ROUTINES::SOLVER_LINEAR_INITIALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR-
ITERATIVE_ABSOLUTE_TOLERANCE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR-
ITERATIVE_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE-
DIVERGENCE_TOLERANCE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE-
FINALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(),   SOLVER-
ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(),   SOLVER-
ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(),   SOLVER-
ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(),   SOLVER-
ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(),   SOLVER_ROUTINES::SOLVER-
LINEAR_ITERATIVE_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(),
SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET(),   SOLVER_ROUTINES::SOLVER-
MATRICES_ASSEMBLE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE-
FINISH(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY(),   SOLVER_MATRICES-
ROUTINES::SOLVER_MATRICES_FINALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER-
MATRICES_INITIALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY-
TYPE_SET(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT(),   SOLVER-
MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(),   SOLVER_MATRICES-
ROUTINES::SOLVER_MATRIX_FINALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER-
MATRIX_INITIALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE-
CALCULATE(),   SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH(),   SOLVER-
ROUTINES::SOLVER_NONLINEAR_FINALISE(),   SOLVER_ROUTINES::SOLVER-
NONLINEAR_INITIALISE(),   SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(),   SOLVER-
ROUTINES::SOLVER_OUTPUT_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_SOLVE(),
SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_TIME-
INTEGRATION_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION-
FINALISE(),   SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(),   SOLVER-
ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE(),   SOLVER_ROUTINES::SOLVER-
VARIABLES_UPDATE(),   STRINGS::STRING_TO_DOUBLE_C(),   STRINGS::STRING-
TO_DOUBLE_VS(),   STRINGS::STRING_TO_INTEGER_C(),   STRINGS::STRING_TO_-
```

INTEGER_VS(), STRINGS::STRING_TO_LOGICAL_C(), STRINGS::STRING_TO_LOGICAL_VS(), STRINGS::STRING_TO_LONG_INTEGER_C(), STRINGS::STRING_TO_LONG_INTEGER_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_REALS_VS(), STRINGS::STRING_TO_SINGLE_C(), STRINGS::STRING_TO_SINGLE_VS(), TREES::TREE_CREATE_FINISH(), TREES::TREE_CREATE_START(), TREES::TREE_DESTROY(), TREES::TREE_DETACH_AND_DESTROY(), TREES::TREE_DETACH_IN_ORDER(), TREES::TREE_FINALISE(), TREES::TREE_INITIALISE(), TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_DELETE(), TREES::TREE_ITEM_INSERT(), TREES::TREE_NODE_FINALISE(), TREES::TREE_NODE_INITIALISE(), TREES::TREE_NODE_KEY_GET(), TREES::TREE_NODE_VALUE_GET(), TREES::TREE_NODE_VALUE_SET(), TREES::TREE_OUTPUT(), TREES::TREE_OUTPUT_IN_ORDER(), TREES::TREE_PREDECESSOR(), TREES::TREE_SEARCH(), TREES::TREE_SUCCESSOR(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(), MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_DP(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_GET_L(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_TYPE_SET(), MATRIX_VECTOR::VECTOR_DESTROY(), MATRIX_VECTOR::VECTOR_DUPLICATE(), MATRIX_VECTOR::VECTOR_FINALISE(), MATRIX_VECTOR::VECTOR_INITIALISE(), MATRIX_VECTOR::VECTOR_SIZE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_L(), MATRIX_VECTOR::VECTOR_VALUES_GET_L1(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP1(), BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(), BINARY_FILE::WRITE_BINARY_FILE_DP(), BINARY_FILE::WRITE_BINARY_FILE_DP1(), BINARY_FILE::WRITE_BINARY_FILE_DPC(), BINARY_FILE::WRITE_BINARY_FILE_DPC1(), BINARY_FILE::WRITE_BINARY_FILE_INTG(), BINARY_FILE::WRITE_BINARY_FILE_INTG1(), BINARY_FILE::WRITE_BINARY_FILE_LINTG(), BINARY_FILE::WRITE_BINARY_FILE_LINTG1(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1(), BINARY_FILE::WRITE_BINARY_FILE_SINTG(), BINARY_FILE::WRITE_BINARY_FILE_SINTG1(), BINARY_FILE::WRITE_BINARY_FILE_SP(), BINARY_FILE::WRITE_BINARY_FILE_SP1(), BINARY_FILE::WRITE_BINARY_FILE_SPC(), BINARY_FILE::WRITE_BINARY_FILE_SPC1(), and BINARY_FILE::WRITE_BINARY_TAG_HEADER().

6.1.2.6 subroutine BASE_ROUTINES::ERRORS (CHARACTER(LEN=*),intent(in) NAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(inout) ERROR)

Records the exiting error of the subroutine.

Parameters:

NAME The name of the routine with an error condition

ERR The error code

ERROR The error string

Definition at line 334 of file base_routines.f90.

References MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_-CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_-SETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_-TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARY_-FILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONAL_-ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_-ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_-NODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_-TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_-TYPE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_-NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_-FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_-RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_-ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_-ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_-INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_-CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_-START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_-SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(), COORDINATE_-ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_-TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_-NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_-ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_-SYSTEMS_INITIALISE(), TIMER::CPU_TIMER(), MATHS::CROSS_PRODUCT_DP(), MATHS::CROSS_PRODUCT_INTG(), MATHS::CROSS_PRODUCT_SP(), COORDINATE_-ROUTINES::D2ZX_DP(), MATHS::D_CROSS_PRODUCT_DP(), MATHS::D_CROSS_PRODUCT_-INTG(), MATHS::D_CROSS_PRODUCT_SP(), MESH_ROUTINES::DECOMPOSITION_-CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_-ROUTINES::DECOMPOSITION_DESTROY(), MESH_ROUTINES::DECOMPOSITION_-ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_-DOMAIN_SET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_-NUMBER_SET(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_-

```

SET(),      MATHS::DETERMINANT_FULL_DP(),      MATHS::DETERMINANT_FULL_INTG(),
MATHS::DETERMINANT_FULL_SP(),          DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_FINALISE(),          DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_INITIALISE(),        DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
LOCAL_FROM_GLOBAL_CALCULATE(),     DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
MAPPING_FINALISE(),                DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_FINALISE(),                 DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_INITIALISE(),               DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
INITIALISE(),         MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(),    MESH_-
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(),   MESH_ROUTINES::DOMAIN_-
TOPOLOGY_CALCULATE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_ELEMENT_FINALISE(),       MESH_ROUTINES::DOMAIN_TOPOLOGY_-
ELEMENT_INITIALISE(),            MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-
FINALISE(),          MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(),      MESH_ROUTINES::DOMAIN_-
TOPOLOGY_INITIALISE(),           MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_-
FROM_MESH(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(),    MESH_-
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(),     MESH_ROUTINES::DOMAIN_-
TOPOLOGY_LINES_FINALISE(),        MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_-
INITIALISE(),        MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(),    MESH_-
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(),     MESH_ROUTINES::DOMAIN_-
TOPOLOGY_NODES_FINALISE(),       MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_-
INITIALISE(),        MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_-
ELEMENTS_CALCULATE(),          COORDINATE_ROUTINES::DXZ_DP(),      COORDINATE_-
ROUTINES::DZX_DP(),      MATHS::EIGENVALUE_FULL_DP(),      MATHS::EIGENVALUE_-
FULL_SP(),      MATHS::EIGENVECTOR_FULL_DP(),      MATHS::EIGENVECTOR_FULL_-
SP(),      EQUATIONS_SET_ROUTINES::EQ(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
INTERPOLATION_FINALISE(), EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_-
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-_
FINISH(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-_
START(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_FINALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_ANALYTIC_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_ASSEMBLE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_ASSEMBLE_LINEAR_STATIC_FEM(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
BACKSUBSTITUTE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_DEPENDENT_(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_FINISH(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_START(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
DEPENDENT_DEPENDENT_FIELD_GET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_DESTROY(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-_
DEPENDENT_FINALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_-_
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_SCALING_-
SET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DESTROY(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_-

```

```

INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_-
TYPE_SET(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_SPARSITY_-
TYPE_SET(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_-
DATA_FINALISE(),   EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_EQUATIONS_-
TIME_DATA_INITIALISE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_EQUATIONS_-
EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_GEOMETRY_FINALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_GEOMETRY_INITIALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_INITIALISE(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SET_MATERIALS_COMPONENT_INTERPOLATION_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_-
SET(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-
FINISH(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-
START(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_FINALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_INITIALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_SCALING_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SETUP(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SET_SOURCE_CREATE_FINISH(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SOURCE_CREATE_START(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_-
DESTROY(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_INITIALISE(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SOURCE_SCALING_SET(),    EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SPECIFICAT(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SET_USER_NUMBER_FIND(),    EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE(),    F90C::F2CSTRING(),
FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(),    FINITE_-
ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_INITIALISE(),    FIELD_ROUTINES::FI(),
FIELD_IO_ROUTINES::FIE(),    FIELD_ROUTINES::FIELD_COMPONENT_INTER(),    FIELD_-
ROUTINES::FIELD_COMPONENT_MESH_COM(),    FIELD_ROUTINES::FIELD_CREATE_-
FINISH(),    FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(),    FIELD_-
ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(),    FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_SET_NUMBER(),    FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_-
PTR(),    FIELD_ROUTINES::FIELD_DESTROY(),    FIELD_ROUTINES::FIELD_DIMENSION_SET_-
NUMBER(),    FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR(),    FIELD_ROUTINES::FIELD_-
INTERPOLATE_GAUSS(),    FIELD_ROUTINES::FIELD_INTERPOLATE_XI(),    FIELD_-
ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(),    FIELD_ROUTINES::FIELD_-
INTERPOLATED_POINT_INITIALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-
METRICS_CALCULATE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_-
FINALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),    FIELD_-
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(),    FIELD_ROUTINES::FIELD_-
INTERPOLATION_PARAMETERS_INITIALISE(),    FIELD_ROUTINES::FIELD_INTERPOLATION_-
PARAMETERS_LINE_GET(),    FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_-
LABEL(),    FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(),    FIELD_IO_ROUTINES::FIELD_-

```

```

IO_DERIVATIVE_INFO(),      FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
ATTACH_LOCAL_PROCESS(),    FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD_-
IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT(),   FIELD_IO_ROUTINES::FIELD_-
IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_-
HEADER_FORTRAN(),        FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_0,
FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA(),   FIELD_-
IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(),  FIELD_IO_ROUTINES::FIELD_IO_-
FIELD_INFO(),   FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET(),  FIELD_-
IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(),      FIELD_IO_ROUTINES::FIELD_IO_FILL_-
BASIS_INFO(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE(),   FIELD_-
IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN(),      FIELD_IO_ROUTINES::FIELD_IO_-
FORTRAN_FILE_READ_DP(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_-
INTG(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING(),  FIELD_IO_-
ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND(),       FIELD_IO_ROUTINES::FIELD_IO_-
FORTRAN_FILE_WRITE_DP(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_-
INTG(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING(),  FIELD_-
IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(),     FIELD_IO_ROUTINES::FIELD_-
IO_LABEL_DERIVATIVE_INFO_GET(),   FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_-
INFO_GET(),   FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET(),  FIELD_-
IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(),  FIELD_-
ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(),     FIELD_IO_ROUTINES::FIELD_-
IO_NODAL_INFO_SET_INITIALISE(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_-
INFO_SET_SORT(),   FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT(),   FIELD_-
ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE(),  FIELD_-
ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE(),   FIELD_ROUTINES::FIELD_-
VARIABLE_COMPONENT_INITIALISE(),  FIELD_ROUTINES::FIELD_VARIABLES_FINALISE(),
FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE(),  FIELD_ROUTINES::FIELDS_FINALISE(),
FIELD_ROUTINES::FIELDS_INITIALISE(),   GENERATED_MESH_ROUTINES::GENERATED_-
MESH_CREATE_FINISH(),      GENERATED_MESH_ROUTINES::GENERATED_MESH_-
CREATE_START(),          GENERATED_MESH_ROUTINES::GENERATED_MESH_-
DESTROY(),   GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(),
GENERATED_MESH_ROUTINES::GENERATED_MESH_INITALISE(),   GENERATED_-
MESH_ROUTINES::GENERATED_MESH_TYPE_SET(),      SORTING::HEAP_SORT_DP(),
SORTING::HEAP_SORT_INTG(),   SORTING::HEAP_SORT_SP(),  BINARY_FILE::INQUIRE_-
EOF_BINARY_FILE(), MATHS::INVERT_FULL_DP(), MATHS::INVERT_FULL_SP(), LAPLACE_-
EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_-_
CALCULATE(),   LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_-
SET_SETUP(),   LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_-
SET_STANDARD_SETUP(),   LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
EQUATIONS_SET_SUBTYPE_SET(),   LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
EQUATION_PROBLEM_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-_
PROBLEM_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-_
PROBLEM_SUBTYPE_SET(),  LISTS::LIST_CREATE_FINISH(),  LISTS::LIST_CREATE_START(),
LISTS::LIST_DATA_TYPE_SET(),  LISTS::LIST_DESTROY(),   LISTS::LIST_DETACH_AND_-
DESTROY_DP(), LISTS::LIST_DETACH_AND_DESTROY_INTG(), LISTS::LIST_DETACH_AND_-
DESTROY_SP(),  LISTS::LIST_FINALISE(),  LISTS::LIST_INITIAL_SIZE_SET(),  LISTS::LIST_-
INITIALISE(), LISTS::LIST_ITEM_ADD_DP1(), LISTS::LIST_ITEM_ADD_INTG1(), LISTS::LIST_-
ITEM_ADD_SP1(),   LISTS::LIST_ITEM_DELETE(),   LISTS::LIST_ITEM_IN_LIST_DP1(),
LISTS::LIST_ITEM_IN_LIST_INTG1(),  LISTS::LIST_ITEM_IN_LIST_SP1(),  LISTS::LIST_-
NUMBER_OF_ITEMS_GET(), LISTS::LIST_REMOVE_DUPLICATES(), LISTS::LIST_SEARCH_-
DP_ARRAY(),  LISTS::LIST_SEARCH_INTG_ARRAY(),  LISTS::LIST_SEARCH_LINEAR_DP_-
ARRAY(),  LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(), LISTS::LIST_SEARCH_LINEAR_-
SP_ARRAY(), LISTS::LIST_SEARCH_SP_ARRAY(), LISTS::LIST_SORT_BUBBLE_DP_ARRAY(),

```

```
LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(),      LISTS::LIST_SORT_BUBBLE_SP_ARRAY(),
LISTS::LIST_SORT_DP_ARRAY(),      LISTS::LIST_SORT_HEAP_DP_ARRAY(),      LISTS::LIST_
SORT_HEAP_INTG_ARRAY(),      LISTS::LIST_SORT_HEAP_SP_ARRAY(),      LISTS::LIST_
SORT_INTG_ARRAY(),      LISTS::LIST_SORT_SHELL_DP_ARRAY(),      LISTS::LIST_SORT_
SHELL_INTG_ARRAY(),      LISTS::LIST_SORT_SHELL_SP_ARRAY(),      LISTS::LIST_SORT_
SP_ARRAY(),      STRINGS::LIST_TO_CHARACTER_C(),      STRINGS::LIST_TO_CHARACTER_
DP(),      STRINGS::LIST_TO_CHARACTER_INTG(),      STRINGS::LIST_TO_CHARACTER_
L(),      STRINGS::LIST_TO_CHARACTER_LINTG(),      STRINGS::LIST_TO_CHARACTER_
SP(),      STRINGS::LOGICAL_TO_CHARACTER(),      STRINGS::LOGICAL_TO_VSTRING(),
MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(),      MATRIX_VECTOR::MATRIX_ALL_
VALUES_SET_INTG(),      MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(),      MATRIX_
VECTOR::MATRIX_ALL_VALUES_SET_SP(),      MATRIX_VECTOR::MATRIX_CREATE_FINISH(),
MATRIX_VECTOR::MATRIX_CREATE_START(),      MATRIX_VECTOR::MATRIX_DATA_GET_
DP(),      MATRIX_VECTOR::MATRIX_DATA_GET_INTG(),      MATRIX_VECTOR::MATRIX_DATA_
GET_L(),      MATRIX_VECTOR::MATRIX_DATA_GET_SP(),      MATRIX_VECTOR::MATRIX_DATA_
TYPE_SET(),      MATRIX_VECTOR::MATRIX_DESTROY(),      MATRIX_VECTOR::MATRIX_
DUPLICATE(),      MATRIX_VECTOR::MATRIX_FINALISE(),      MATRIX_VECTOR::MATRIX_
INITIALISE(),      MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(),      MATRIX_
VECTOR::MATRIX_MAX_SIZE_SET(),      MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_
SET(),      MATRIX_VECTOR::MATRIX_OUTPUT(),      MATRIX_VECTOR::MATRIX_SIZE_SET(),
MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(),      MATRIX_VECTOR::MATRIX_
STORAGE_LOCATIONS_GET(),      MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_
SET(),      MATRIX_VECTOR::MATRIX_STORAGE_TYPE_GET(),      MATRIX_VECTOR::MATRIX_
STORAGE_TYPE_SET(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(),      MATRIX_
VECTOR::MATRIX_VALUES_ADD_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_
DP2(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG(),      MATRIX_VECTOR::MATRIX_
VALUES_ADD_INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(),      MATRIX_
VECTOR::MATRIX_VALUES_ADD_L(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_L1(),
MATRIX_VECTOR::MATRIX_VALUES_ADD_L2(),      MATRIX_VECTOR::MATRIX_VALUES_
ADD_SP(),      MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(),      MATRIX_VECTOR::MATRIX_
VALUES_ADD_SP2(),      MATRIX_VECTOR::MATRIX_VALUES_GET_DP(),      MATRIX_
VECTOR::MATRIX_VALUES_GET_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_GET_
DP2(),      MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(),      MATRIX_VECTOR::MATRIX_
VALUES_GET_INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(),      MATRIX_
VECTOR::MATRIX_VALUES_GET_L(),      MATRIX_VECTOR::MATRIX_VALUES_GET_L1(),
MATRIX_VECTOR::MATRIX_VALUES_GET_L2(),      MATRIX_VECTOR::MATRIX_VALUES_
GET_SP(),      MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(),      MATRIX_VECTOR::MATRIX_
VALUES_GET_SP2(),      MATRIX_VECTOR::MATRIX_VALUES_SET_DP(),      MATRIX_
VECTOR::MATRIX_VALUES_SET_DP1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_
DP2(),      MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(),      MATRIX_VECTOR::MATRIX_
VALUES_SET_INTG1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(),      MATRIX_
VECTOR::MATRIX_VALUES_SET_L(),      MATRIX_VECTOR::MATRIX_VALUES_SET_
L1(),      MATRIX_VECTOR::MATRIX_VALUES_SET_L2(),      MATRIX_VECTOR::MATRIX_
VALUES_SET_SP(),      MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(),      MATRIX_
VECTOR::MATRIX_VALUES_SET_SP2(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_
ROUTINES::MESH_CREATE_REGULAR(),      MESH_ROUTINES::MESH_CREATE_START(),
MESH_ROUTINES::MESH_DESTROY(),      MESH_ROUTINES::MESH_FINALISE(),      MESH_
ROUTINES::MESH_INITIALISE(),      MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_
SET_NUMBER(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(), MESH_
ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(),      MESH_ROUTINES::MESH_
NUMBER_OF_ELEMENTS_SET_PTR(), MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(),
MESH_ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(),      MESH_ROUTINES::MESH_
TOPOLOGY_DOFS_FINALISE(),      MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(),
MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(),      MESH_ROUTINES::MESH_
```

TOPOLOGY_ELEMENT_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
 ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
 BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_-
 ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_-
 TOPOLOGY_ELEMENTS_DESTROY(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-
 ELEMENT_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_-
 NODES_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(), MESH_-
 ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(), MESH_ROUTINES::MESH_-
 TOPOLOGY_ELEMENTS_NUMBER_SET(), MESH_ROUTINES::MESH_TOPOLOGY_-
 FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(), MESH_ROUTINES::MESH_-
 TOPOLOGY_NODE_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(),
 MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_-
 TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_-
 NODES_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(), MESH_-
 ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE(),
 MESH_ROUTINES::MESH_USER_NUMBER_FIND(), MESH_ROUTINES::MESSES_FINALISE(),
 MESH_ROUTINES::MESSES_INITIALISE(), CMISS_MPI::MPI_ERROR_CHECK(), NODE_-
 ROUTINES::NODE_CHECK_EXISTS(), NODE_ROUTINES::NODE_DESTROY(), NODE_-
 ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_ROUTINES::NODE_NUMBER_SET(),
 NODE_ROUTINES::NODES_CREATE_FINISH(), NODE_ROUTINES::NODES_CREATE_-
 START(), NODE_ROUTINES::NODES_FINALISE(), NODE_ROUTINES::NODES_INITIALISE(),
 MATHS::NORMALISE_DP(), MATHS::NORMALISE_SP(), STRINGS::NUMBER_TO_-
 CHARACTER_DP(), STRINGS::NUMBER_TO_CHARACTER_INTG(), STRINGS::NUMBER_-
 TO_CHARACTER_LINTG(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_-
 TO_VSTRING_DP(), STRINGS::NUMBER_TO_VSTRING_INTG(), STRINGS::NUMBER_TO_-
 VSTRING_LINTG(), STRINGS::NUMBER_TO_VSTRING_SP(), BINARY_FILE::OPEN_BINARY_-
 FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_PARMETIS::PARMETIS_-
 PARTKWAY(), CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), CMISS_PETSC::PETSC_-
 ERRORHANDLING_SET_OFF(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON(),
 CMISS_PETSC::PETSC_FINALIZE(), CMISS_PETSC::PETSC_INITIALIZE(), CMISS_-
 PETSC::PETSC_ISDESTROY(), CMISS_PETSC::PETSC_ISFINALISE(), CMISS_PETSC::PETSC_-
 ISINITIALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(),
 CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(), CMISS_-
 PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(), CMISS_PETSC::PETSC_-
 ISLOCALTOGLOBALMAPPINGDESTROY(), CMISS_PETSC::PETSC_-
 ISLOCALTOGLOBALMAPPINGFINALISE(), CMISS_PETSC::PETSC_-
 ISLOCALTOGLOBALMAPPINGINITIALISE(), CMISS_PETSC::PETSC_KSPCREATE(),
 CMISS_PETSC::PETSC_KSPDESTROY(), CMISS_PETSC::PETSC_KSPFINALISE(),
 CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(), CMISS_PETSC::PETSC_-
 KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_-
 PETSC::PETSC_KSPGETRESIDUALNORM(), CMISS_PETSC::PETSC_KSPINITIALISE(),
 CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(), CMISS_PETSC::PETSC_-
 KSPSETOPERATORS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(), CMISS_-
 PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PETSC_KSPSETUP(), CMISS_PETSC::PETSC_-
 KSPSOLVE(), CMISS_PETSC::PETSC_LOGPRINTSUMMARY(), CMISS_PETSC::PETSC_-
 MATASSEMBLYBEGIN(), CMISS_PETSC::PETSC_MATASSEMBLYEND(), CMISS_-
 PETSC::PETSC_MATCREATE(), CMISS_PETSC::PETSC_MATCREATEMPIAIJ(), CMISS_-
 PETSC::PETSC_MATCREATEMPIENSE(), CMISS_PETSC::PETSC_MATCREATESEQAIJ(),
 CMISS_PETSC::PETSC_MATCREATESEQDENSE(), CMISS_PETSC::PETSC_MATDESTROY(),
 CMISS_PETSC::PETSC_MATFINALISE(), CMISS_PETSC::PETSC_MATGETARRAY(),
 CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_-
 MATGETVALUES(), CMISS_PETSC::PETSC_MATINITIALISE(), CMISS_PETSC::PETSC_-
 MATRESTOREARRAY(), CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(),
 CMISS_PETSC::PETSC_MATSETOPTION(), CMISS_PETSC::PETSC_MATSETSIZES(), CMISS_-

PETSC::PETSC_MATSETVALUE(), CMISS_PETSC::PETSC_MATSETVALUELOCAL(), CMISS_PETSC::PETSC_MATSETVALUES(), CMISS_PETSC::PETSC_MATSETVALUESLOCAL(),
 CMISS_PETSC::PETSC_MATVIEW(), CMISS_PETSC::PETSC_MATZEROENTRIES(),
 CMISS_PETSC::PETSC_PCFINALISE(), CMISS_PETSC::PETSC_PCINITIALISE(), CMISS_PETSC::PETSC_PCSETTYPE(), CMISS_PETSC::PETSC_VECASSEMBLYBEGIN(),
 CMISS_PETSC::PETSC_VECASSEMBLYEND(), CMISS_PETSC::PETSC_VECCREATE(), CMISS_PETSC::PETSC_VECCREATEGHOST(), CMISS_PETSC::PETSC_VECCREATEGHOSTWITHARRAY(),
 CMISS_PETSC::PETSC_VECCREATEMPI(), CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY(), CMISS_PETSC::PETSC_VECCREATESEQ(),
 CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY(), CMISS_PETSC::PETSC_VECDESTROY(), CMISS_PETSC::PETSC_VECDUPLICATE(),
 CMISS_PETSC::PETSC_VECFINALISE(), CMISS_PETSC::PETSC_VECGETARRAY(), CMISS_PETSC::PETSC_VECGETARRAYF90(),
 CMISS_PETSC::PETSC_VECGETLOCALSIZE(), CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_VECGETSIZE(),
 PETSC::PETSC_VECGETVALUES(), CMISS_PETSC::PETSC_VECHOSTGETLOCALFORM(), CMISS_PETSC::PETSC_VECGHOSTRESTORELOCALFORM(),
 CMISS_PETSC::PETSC_VECGHOSTUPDATEBEGIN(), CMISS_PETSC::PETSC_VECGHOSTUPDATEEND(), CMISS_PETSC::PETSC_VECINITIALISE(),
 CMISS_PETSC::PETSC_VCRESTOREARRAY(), CMISS_PETSC::PETSC_VCRESTOREARRAYF90(), CMISS_PETSC::PETSC_VECSET(),
 CMISS_PETSC::PETSC_VECSETFROMOPTIONS(), CMISS_PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING(), CMISS_PETSC::PETSC_VECSETSIZES(),
 PETSC::PETSC_VECSETVALUES(), CMISS_PETSC::PETSC_VECSETVALUESLOCAL(), CMISS_PETSC::PETSC_VECVIEW(),
 PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_CONTROL_DESTROY(), PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE(),
 PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER(), PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR(),
 PROBLEM_ROUTINES::PROBLEM_FINALISE(), PROBLEM_ROUTINES::PROBLEM_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SETUP(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_FINALISE(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTION_INITIALISE(), PROBLEM_ROUTINES::PROBLEM_SOLUTION_solve(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(),
 PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE(), PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(),
 PROBLEM_ROUTINES::PROBLEM_SOLVE(), PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH(),
 PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(), PROBLEM_ROUTINES::PROBLEM_SOLVER_GET(),
 PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR(),
 PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND(), PROBLEMS_FINALISE(), PROBLEM_ROUTINES::PROBLEMS_INITIALISE(),
 BINARY_FILE::READ_BINARY_FILE_CHARACTER(), BINARY_FILE::READ_BINARY_FILE_DP(), BINARY_FILE::READ_BINARY_FILE_DP1(),
 BINARY_FILE::READ_BINARY_FILE_DPC(), BINARY_FILE::READ_BINARY_FILE_DPC1(), BINARY_FILE::READ_BINARY_FILE_INTG(),
 BINARY_FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_FILE::READ_BINARY_FILE_LINTG1(),
 BINARY_FILE::READ_BINARY_FILE_LOGICAL(), BINARY_FILE::READ_BINARY_FILE_LOGICAL1(), BINARY_FILE::READ_BINARY_FILE_SINTG(),
 BINARY_FILE::READ_BINARY_FILE_SINTG1(), BINARY_FILE::READ_BINARY_FILE_SP(), BINARY_FILE::READ_BINARY_FILE_SP1(),
 BINARY_FILE::READ_BINARY_FILE_SPC(), BINARY_FILE::READ_BINARY_FILE_SPC1(),
 BINARY_FILE::READ_BINARY_TAG_HEADER(), REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET(),
 REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET()

```

SET_NUMBER(),           REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR(),
REGION_ROUTINES::REGION_CREATE_FINISH(),   REGION_ROUTINES::REGION_CREATE_
-START(),   REGION_ROUTINES::REGION_DESTROY(),   REGION_ROUTINES::REGION_
-LABEL_GET(),   REGION_ROUTINES::REGION_LABEL_SET_NUMBER(),   REGION_
-ROUTINES::REGION_LABEL_SET_PTR(),   REGION_ROUTINES::REGION_SUB_REGION_
-CREATE_FINISH(),   REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(),
REGION_ROUTINES::REGION_USER_NUMBER_FIND(),   REGION_ROUTINES::REGION_
-USER_NUMBER_FIND_PTR(),   REGION_ROUTINES::REGIONS_FINALISE(),   REGION_
-ROUTINES::REGIONS_INITIALISE(),   BINARY_FILE::RESET_BINARY_NUMBER_TAGS(),
BINARY_FILE::SET_BINARY_FILE(),   SORTING::SHELL_SORT_DP(),   SORTING::SHELL_
-SORT_INTG(),   SORTING::SHELL_SORT_SP(),   BINARY_FILE::SKIP_BINARY_FILE(),   BINARY_
-FILE::SKIP_BINARY_TAGS(),   BINARY_FILE::SKIP_CM_BINARY_HEADER(),   MATHS::SOLVE_
-SMALL_LINEAR_SYSTEM_DP(),   MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(),   SOLVER_
-ROUTINES::SOLVER_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_CREATE_
-START(),   SOLVER_ROUTINES::SOLVER_DESTROY(),   SOLVER_ROUTINES::SOLVER_
-EIGENPROBLEM_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_
-FINALISE(),   SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(),   SOLVER_
-ROUTINES::SOLVER_EIGENPROBLEM_SOLVE(),   SOLVER_ROUTINES::SOLVER_FINALISE(),
SOLVER_ROUTINES::SOLVER_INITIALISE(),   SOLVER_ROUTINES::SOLVER_LIBRARY_SET(),
SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_
-LINEAR_DIRECT_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_
-FINALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(),   SOLVER_ROUTINES::SOLVER_LINEAR_
-DIRECT_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_INITIALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR_
-ITERATIVE_ABSOLUTE_TOLERANCE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_
-ITERATIVE_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_
-DIVERGENCE_TOLERANCE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_
-FINALISE(),   SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(),   SOLVER_
-ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(),   SOLVER_ROUTINES::SOLVER_
-LINEAR_ITERATIVE_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(),
SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_
-MATRICES_ASSEMBLE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_
-FINISH(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),
SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY(),   SOLVER_MATRICES_
-ROUTINES::SOLVER_MATRICES_FINALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_
-MATRICES_INITIALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_
-TYPE_SET(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT(),   SOLVER_
-MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(),   SOLVER_MATRICES_
-ROUTINES::SOLVER_MATRIX_FINALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_
-MATRIX_INITIALISE(),   SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_
-CALCULATE(),   SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH(),   SOLVER_
-ROUTINES::SOLVER_NONLINEAR_FINALISE(),   SOLVER_ROUTINES::SOLVER_
-NONLINEAR_INITIALISE(),   SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(),   SOLVER_
-ROUTINES::SOLVER_OUTPUT_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_SOLVE(),
SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET(),   SOLVER_ROUTINES::SOLVER_TIME_
-INTEGRATION_CREATE_FINISH(),   SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_
-FINALISE(),   SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(),   SOLVER_
-ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE(),   SOLVER_ROUTINES::SOLVER_
-VARIABLES_UPDATE(),   STRINGS::STRING_TO_DOUBLE_C(),   STRINGS::STRING_
-TO_DOUBLE_VS(),   STRINGS::STRING_TO_INTEGER_C(),   STRINGS::STRING_TO_

```

```

INTEGER_VS(),   STRINGS::STRING_TO_LOGICAL_C(),   STRINGS::STRING_TO_LOGICAL_-
VS(),   STRINGS::STRING_TO_LONG_INTEGER_C(),   STRINGS::STRING_TO_LONG_-
INTEGER_VS(),   FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(),   FIELD_-
IO_ROUTINES::STRING_TO_MUTI_REALS_VS(),   STRINGS::STRING_TO_SINGLE_C(),
STRINGS::STRING_TO_SINGLE_VS(),   TREES::TREE_CREATE_FINISH(),   TREES::TREE_-
CREATE_START(),   TREES::TREE_DESTROY(),   TREES::TREE_DETACH_AND_DESTROY(),
TREES::TREE_DETACH_IN_ORDER(),   TREES::TREE_FINALISE(),   TREES::TREE_INITIALISE(),
TREES::TREE_INSERT_TYPE_SET(),   TREES::TREE_ITEM_DELETE(),   TREES::TREE_-
ITEM_INSERT(),   TREES::TREE_NODE_FINALISE(),   TREES::TREE_NODE_INITIALISE(),
TREES::TREE_NODE_KEY_GET(),   TREES::TREE_NODE_VALUE_GET(),   TREES::TREE_-
NODE_VALUE_SET(),   TREES::TREE_OUTPUT(),   TREES::TREE_OUTPUT_IN_ORDER(),
TREES::TREE_PREDECESSOR(),   TREES::TREE_SEARCH(),   TREES::TREE_SUCCESSOR(),
MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(),   MATRIX_VECTOR::VECTOR_ALL_-
VALUES_SET_INTG(),   MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(),   MATRIX_-
VECTOR::VECTOR_ALL_VALUES_SET_SP(),   MATRIX_VECTOR::VECTOR_CREATE_FINISH(),
MATRIX_VECTOR::VECTOR_CREATE_START(),   MATRIX_VECTOR::VECTOR_DATA_GET_-
DP(),   MATRIX_VECTOR::VECTOR_DATA_GET_INTG(),   MATRIX_VECTOR::VECTOR_DATA_-
GET_L(),   MATRIX_VECTOR::VECTOR_DATA_GET_SP(),   MATRIX_VECTOR::VECTOR_DATA_-
TYPE_SET(),   MATRIX_VECTOR::VECTOR_DESTROY(),   MATRIX_VECTOR::VECTOR_-
DUPLICATE(),   MATRIX_VECTOR::VECTOR_FINALISE(),   MATRIX_VECTOR::VECTOR_-
INITIALISE(),   MATRIX_VECTOR::VECTOR_SIZE_SET(),   MATRIX_VECTOR::VECTOR_-
VALUES_GET_DP(),   MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(),   MATRIX_-
VECTOR::VECTOR_VALUES_GET_INTG(),   MATRIX_VECTOR::VECTOR_VALUES_GET_-
INTG1(),   MATRIX_VECTOR::VECTOR_VALUES_GET_L(),   MATRIX_VECTOR::VECTOR_-
VALUES_GET_L1(),   MATRIX_VECTOR::VECTOR_VALUES_GET_SP(),   MATRIX_-
VECTOR::VECTOR_VALUES_GET_SP1(),   MATRIX_VECTOR::VECTOR_VALUES_SET_-
DP(),   MATRIX_VECTOR::VECTOR_VALUES_SET_DP1(),   MATRIX_VECTOR::VECTOR_-
VALUES_SET_INTG(),   MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1(),   MATRIX_-
VECTOR::VECTOR_VALUES_SET_L(),   MATRIX_VECTOR::VECTOR_VALUES_SET_L1(),
MATRIX_VECTOR::VECTOR_VALUES_SET_SP(),   MATRIX_VECTOR::VECTOR_VALUES_-
SET_SP1(),   INPUT_OUTPUT::WR(),   BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(),
BINARY_FILE::WRITE_BINARY_FILE_DP(),   BINARY_FILE::WRITE_BINARY_FILE_DP1(),
BINARY_FILE::WRITE_BINARY_FILE_DPC(),   BINARY_FILE::WRITE_BINARY_FILE_-
DPC1(),   BINARY_FILE::WRITE_BINARY_FILE_INTG(),   BINARY_FILE::WRITE_BINARY_-
FILE_INTG1(),   BINARY_FILE::WRITE_BINARY_FILE_LINTG(),   BINARY_FILE::WRITE_-
BINARY_FILE_LINTG1(),   BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(),   BINARY_-
FILE::WRITE_BINARY_FILE_LOGICAL1(),   BINARY_FILE::WRITE_BINARY_FILE_SINTG(),
BINARY_FILE::WRITE_BINARY_FILE_SINTG1(),   BINARY_FILE::WRITE_BINARY_FILE_-
SP(),   BINARY_FILE::WRITE_BINARY_FILE_SP1(),   BINARY_FILE::WRITE_BINARY_FILE_-
SPC(),   BINARY_FILE::WRITE_BINARY_FILE_SPC1(),   BINARY_FILE::WRITE_BINARY_TAG_-
HEADER(),   INPUT_OUTPUT::WRITE_STRING_C(),   INPUT_OUTPUT::WRITE_STRING_-
FMT(),   INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C(),   INPUT_OUTPUT::WRITE_-
STRING_FMT_VALUE_DP(),   INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG(),
INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L(),   INPUT_OUTPUT::WRITE_STRING_-
FMT_VALUE_LINTG(),   INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP(),   INPUT_-
OUTPUT::WRITE_STRING_FMT_VALUE_VS(),   INPUT_OUTPUT::WRITE_STRING_IDX(),
INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(),   INPUT_OUTPUT::WRITE_STRING_-
MATRIX_INTG(),   INPUT_OUTPUT::WRITE_STRING_MATRIX_L(),   INPUT_OUTPUT::WRITE_-
STRING_MATRIX_LINTG(),   INPUT_OUTPUT::WRITE_STRING_MATRIX_SP(),   INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_C_C(),   INPUT_OUTPUT::WRITE_STRING_TWO_-
VALUE_C_DP(),   INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG(),   INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_C_L(),   INPUT_OUTPUT::WRITE_STRING_-
TWO_VALUE_C_SP(),   INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS(),   INPUT_-
OUTPUT::WRITE_STRING_TWO_VALUE_DP_C(),   INPUT_OUTPUT::WRITE_STRING_TWO_-

```

VALUE_DP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(), INPUT_OUTPUT::WRITE_STRING_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_VALUE_VS(), and INPUT_OUTPUT::WRITE_STRING_VS().

6.1.2.7 subroutine BASE_ROUTINES::EXITS (CHARACTER(LEN=*)*intent(in)* NAME)

Records the exit out of the named procedure.

See also:

[BASE_ROUTINES::ENTERS](#)

Parameters:

NAME The name of the routine exiting

Definition at line 356 of file base_routines.f90.

Referenced by SORTING::BUBBLE_SORT_DP(), SORTING::BUBBLE_SORT_INTG(), SORTING::BUBBLE_SORT_SP(), F90C::C2FSTRING(), CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITEELEMENTCALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETSETUP(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASSTYPESET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), BINARY_FILE::CLOSE_BINARY_FILE(), BINARY_FILE::CLOSE_CMISS_BINARYFILE(), COORDINATE_ROUTINES::CO(), COMP_ENVIRONMENT::COMPUTATIONALENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONALENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONALNODE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONALNODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONALTYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONALTYPE_INITIALISE(), NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONALNODES_

```

NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_CONVERT_-
FROM_RC_DP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_-
RC_SP(), COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP(),
COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP(), COORDINATE_-
ROUTINES::COORDINATE_DELTA_CALCULATE_DP(), COORDINATE_-
ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_-
INTERPOLATION_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-
INTERPOLATION_PARAMETERS_ADJUST(), COORDINATE_ROUTINES::COORDINATE_-
METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_-
START(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-
PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_-
SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR(),
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR(),
ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_-
TYPE_SET_PTR(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_-
NUMBER(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR(),
COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEMS_FINALISE(), COORDINATE_ROUTINES::COORDINATE_-
SYSTEMS_INITIALISE(), TIMER::CPU_TIMER(), MATHS::CROSS_PRODUCT_DP(),
MATHS::CROSS_PRODUCT_INTG(), MATHS::CROSS_PRODUCT_SP(), COORDINATE_-
ROUTINES::D2ZX_DP(), MATHS::D_CROSS_PRODUCT_DP(), MATHS::D_CROSS_PRODUCT_-
INTG(), MATHS::D_CROSS_PRODUCT_SP(), MESH_ROUTINES::DECOMPOSITION_-
CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_-
ROUTINES::DECOMPOSITION_DESTROY(), MESH_ROUTINES::DECOMPOSITION_-
ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_-
DOMAIN_SET(), MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_-
NUMBER_SET(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_-
SET(), MATHS::DETERMINANT_FULL_DP(), MATHS::DETERMINANT_FULL_INTG(),
MATHS::DETERMINANT_FULL_SP(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
ADJACENT_DOMAIN_INITIALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
LOCAL_FROM_GLOBAL_CALCULATE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_-
MAPPING_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
GLOBAL_INITIALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_-
INITIALISE(), MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(), MESH_-
ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_CALCULATE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_-
ELEMENT_INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-
FINALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(),
MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE(), MESH_ROUTINES::DOMAIN_-
TOPOLOGY_INITIALISE(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_-

```

```

FROM_MESH(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE(),      MESH_-
ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE(),      MESH_ROUTINES::DOMAIN_-
TOPOLOGY_LINES_FINALISE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_-
INITIALISE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE(),      MESH_-
ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE(),      MESH_ROUTINES::DOMAIN_-
TOPOLOGY_NODES_FINALISE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_-
INITIALISE(),      MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_-
ELEMENTS_CALCULATE(),      COORDINATE_ROUTINES::DXZ_DP(),      COORDINATE_-
ROUTINES::DZX_DP(),      MATHS::EIGENVALUE_FULL_DP(),      MATHS::EIGENVALUE_-
FULL_SP(),      MATHS::EIGENVECTOR_FULL_DP(),      MATHS::EIGENVECTOR_FULL_-
SP(),      EQUATIONS_SET_ROUTINES::EQ(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
INTERPOLATION_FINALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_-
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-
FINISH(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_-
START(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_DESTROY(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_ANALYTIC_FINALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_ANALYTIC_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_ASSEMBLE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_ASSEMBLE_LINEAR_STATIC_FEM(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
BACKSUBSTITUTE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_DEPENDENT(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_FINISH(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_CREATE_START(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
DEPENDENT_DEPENDENT_FIELD_GET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-
SET_DEPENDENT_DESTROY(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_-
FINALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_-
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_SCALING_-
SET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DESTROY(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_-
INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_-
TYPE_SET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_SPARSITY_-
TYPE_SET(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_-
DATA_FINALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_-
TIME_DATA_INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FINALISE(),      EQUATIONS_-
SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_DESTROY(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF5(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_GEOMETRY_FINALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_GEOMETRY_INITIALISE(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_INITIALISE(),      EQUATIONS_SET_ROUTINES::EQUATIONS_-

```

```

SET_MATERIALS_COMPONENT_INTERPOLATION_SET(),           EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_-
SET(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-
FINISH(),        EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_-
START(),         EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_DESTROY(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_FINALISE(),   EQUATIONS-
SET_ROUTINES::EQUATIONS_SET_MATERIALS_INITIALISE(),       EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET(),    EQUATIONS-
SET_ROUTINES::EQUATIONS_SET_MATERIALS_SCALING_SET(),      EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SETUP(),           EQUATIONS_SET_ROUTINES::EQUATIONS-
SET_SOURCE_CREATE_FINISH(),      EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
SOURCE_CREATE_START(),          EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_-
DESTROY(),         EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_INITIALISE(),  EQUATIONS-
SET_ROUTINES::EQUATIONS_SET_SOURCE_SCALING_SET(),       EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_SPECIFICAT(),     EQUATIONS_SET_ROUTINES::EQUATIONS-
SET_USER_NUMBER_FIND(),   EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE(),
EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE(),      F90C::F2CSTRING(),
FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_FINALISE(), FINITE-
ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_INITIALISE(),    FIELD_ROUTINES::FI(),
FIELD_IO_ROUTINES::FIE(),      FIELD_ROUTINES::FIELD_COMPONENT_INTER(), FIELD-
ROUTINES::FIELD_COMPONENT_MESH_COM(),      FIELD_ROUTINES::FIELD_CREATE_-
FINISH(),         FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE(), FIELD-
ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE(), FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_SET_NUMBER(),   FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_-
PTR(), FIELD_ROUTINES::FIELD_DESTROY(), FIELD_ROUTINES::FIELD_DIMENSION_SET_-
NUMBER(), FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR(), FIELD_ROUTINES::FIELD_-
INTERPOLATE_GAUSS(),        FIELD_ROUTINES::FIELD_INTERPOLATE_XI(), FIELD-
ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_ROUTINES::FIELD_-
INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-
METRICS_CALCULATE(),        FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_-
FINALISE(),         FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(),
FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), FIELD-
ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE(), FIELD_ROUTINES::FIELD_-
INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_-
PARAMETERS_LINE_GET(),      FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_-
LABEL(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), FIELD_IO_ROUTINES::FIELD_-
IO_DERIVATIVE_INFO(),        FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
ATTACH_LOCAL_PROCESS(),      FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
FINALIZE(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE(), FIELD-
IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT(), FIELD_IO_ROUTINES::FIELD_-
IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_-
HEADER_FORTRAN(),          FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_0,
FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA(), FIELD-
IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC(), FIELD_IO_ROUTINES::FIELD_IO_-
FIELD_INFO(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET(), FIELD-
IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(),   FIELD_IO_ROUTINES::FIELD_IO_FILL_-
BASIS_INFO(),   FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE(), FIELD-
IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN(), FIELD_IO_ROUTINES::FIELD_IO_-
FORTRAN_FILE_READ_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_-
INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING(), FIELD_IO_-
ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND(), FIELD_IO_ROUTINES::FIELD_IO_-
FORTRAN_FILE_WRITE_DP(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_-
INTG(), FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING(), FIELD-

```

```

IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(),           FIELD_IO_ROUTINES::FIELD_-
IO_LABEL_DERIVATIVE_INFO_GET(),          FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_-
INFO_GET(),      FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET(),    FIELD_IO_-
ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS(),    FIELD_IO_-
ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE(),      FIELD_IO_ROUTINES::FIELD_-
IO_NODAL_INFO_SET_INITIALISE(),      FIELD_IO_ROUTINES::FIELD_IO_NODAL_-
INFO_SET_SORT(),      FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT(),    FIELD_IO_-
ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE(),    FIELD_-
ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE(),      FIELD_ROUTINES::FIELD_-
VARIABLE_COMPONENT_INITIALISE(),      FIELD_ROUTINES::FIELD_VARIABLES_FINALISE(), 
FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE(),      FIELD_ROUTINES::FIELDS_FINALISE(), 
FIELD_ROUTINES::FIELDS_INITIALISE(),      GENERATED_MESH_ROUTINES::GENERATED_-
MESH_CREATE_FINISH(),      GENERATED_MESH_ROUTINES::GENERATED_-
MESH_CREATE_START(),      GENERATED_MESH_ROUTINES::GENERATED_MESH_-
DESTROY(),      GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE(), 
GENERATED_MESH_ROUTINES::GENERATED_MESH_INITALISE(),      GENERATED_-
MESH_ROUTINES::GENERATED_MESH_TYPE_SET(),      SORTING::HEAP_SORT_DP(), 
SORTING::HEAP_SORT_INTG(),      SORTING::HEAP_SORT_SP(),    BINARY_FILE::INQUIRE_-
EOF_BINARY_FILE(),    MATHS::INVERT_FULL_DP(),    MATHS::INVERT_FULL_SP(), 
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_-_
CALCULATE(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_-_
SET_SETUP(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_-_
SET_STANDARD_SETUP(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-_
EQUATIONS_SET_SUBTYPE_SET(),      LAPLACE_EQUATIONS_ROUTINES::LAPLACE_-_
EQUATION_PROBLEM_SETUP(),    LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-_
PROBLEM_STANDARD_SETUP(),    LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-_
PROBLEM_SUBTYPE_SET(),    LISTS::LIST_CREATE_FINISH(),    LISTS::LIST_CREATE_START(), 
LISTS::LIST_DATA_TYPE_SET(),      LISTS::LIST_DESTROY(),    LISTS::LIST_DETACH_AND_-_
DESTROY_DP(),    LISTS::LIST_DETACH_AND_DESTROY_INTG(),    LISTS::LIST_DETACH_AND_-_
DESTROY_SP(),    LISTS::LIST_FINALISE(),    LISTS::LIST_INITIAL_SIZE_SET(), 
LISTS::LIST_INITIALISE(),    LISTS::LIST_ITEM_ADD_DP1(),    LISTS::LIST_ITEM_ADD_INTG1(), 
LISTS::LIST_ITEM_ADD_SP1(),      LISTS::LIST_ITEM_DELETE(),    LISTS::LIST_ITEM_IN_LIST_DP1(), 
LISTS::LIST_ITEM_IN_LIST_INTG1(),    LISTS::LIST_ITEM_IN_LIST_SP1(),    LISTS::LIST_-_
NUMBER_OF_ITEMS_GET(),    LISTS::LIST_REMOVE_DUPLICATES(),    LISTS::LIST_SEARCH_-_
DP_ARRAY(),    LISTS::LIST_SEARCH_INTG_ARRAY(),    LISTS::LIST_SEARCH_LINEAR_DP_-_
ARRAY(),    LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY(),    LISTS::LIST_SEARCH_LINEAR_-_
SP_ARRAY(),    LISTS::LIST_SEARCH_SP_ARRAY(),    LISTS::LIST_SORT_BUBBLE_DP_ARRAY(), 
LISTS::LIST_SORT_BUBBLE_INTG_ARRAY(),    LISTS::LIST_SORT_BUBBLE_SP_ARRAY(), 
LISTS::LIST_SORT_DP_ARRAY(),    LISTS::LIST_SORT_HEAP_DP_ARRAY(),    LISTS::LIST_-_
SORT_HEAP_INTG_ARRAY(),    LISTS::LIST_SORT_HEAP_SP_ARRAY(),    LISTS::LIST_-_
SORT_INTG_ARRAY(),    LISTS::LIST_SORT_SHELL_DP_ARRAY(),    LISTS::LIST_SORT_-_
SHELL_INTG_ARRAY(),    LISTS::LIST_SORT_SHELL_SP_ARRAY(),    LISTS::LIST_SORT_-_
SP_ARRAY(),    STRINGS::LIST_TO_CHARACTER_C(),    STRINGS::LIST_TO_CHARACTER_-_
DP(),    STRINGS::LIST_TO_CHARACTER_INTG(),    STRINGS::LIST_TO_CHARACTER_-_
L(),    STRINGS::LIST_TO_CHARACTER_LINTG(),    STRINGS::LIST_TO_CHARACTER_-_
SP(),    STRINGS::LOGICAL_TO_CHARACTER(),    STRINGS::LOGICAL_TO_VSTRING(), 
MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP(),    MATRIX_VECTOR::MATRIX_ALL_-_
VALUES_SET_INTG(),    MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L(),    MATRIX_-_
VECTOR::MATRIX_ALL_VALUES_SET_SP(),    MATRIX_VECTOR::MATRIX_CREATE_FINISH(), 
MATRIX_VECTOR::MATRIX_CREATE_START(),    MATRIX_VECTOR::MATRIX_DATA_GET_-_
DP(),    MATRIX_VECTOR::MATRIX_DATA_GET_INTG(),    MATRIX_VECTOR::MATRIX_DATA_-_
GET_L(),    MATRIX_VECTOR::MATRIX_DATA_GET_SP(),    MATRIX_VECTOR::MATRIX_DATA_-_
TYPE_SET(),    MATRIX_VECTOR::MATRIX_DESTROY(),    MATRIX_VECTOR::MATRIX_-_
DUPLICATE(),    MATRIX_VECTOR::MATRIX_FINALISE(),    MATRIX_VECTOR::MATRIX_-_

```

INITIALISE(), MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET(), MATRIX_VECTOR::MATRIX_MAX_SIZE_SET(), MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_VECTOR::MATRIX_OUTPUT(), MATRIX_VECTOR::MATRIX_SIZE_SET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET(), MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_GET(), MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L10(), MATRIX_VECTOR::MATRIX_VALUES_ADD_L20(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1(), MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_DP2(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_GET_L(), MATRIX_VECTOR::MATRIX_VALUES_GET_L10(), MATRIX_VECTOR::MATRIX_VALUES_GET_L20(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP1(), MATRIX_VECTOR::MATRIX_VALUES_GET_SP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP1(), MATRIX_VECTOR::MATRIX_VALUES_SET_DP2(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1(), MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2(), MATRIX_VECTOR::MATRIX_VALUES_SET_L(), MATRIX_VECTOR::MATRIX_VALUES_SET_L10(), MATRIX_VECTOR::MATRIX_VALUES_SET_L20(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP1(), MATRIX_VECTOR::MATRIX_VALUES_SET_SP2(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_CREATE_REGULAR(), MESH_ROUTINES::MESH_CREATE_START(), MESH_ROUTINES::MESH_DESTROY(), MESH_ROUTINES::MESH_FINALISE(), MESH_ROUTINES::MESH_INITIALISE(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_NUMBER(), MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR(), MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER(), MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_PTR(), MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_DOFS_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(), MESH_ROUTINES::MESH_TOPOLOGY_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_USER_NUMBER_FIND(), MESH_ROUTINES::MESHS_FINALISE(), MESH_ROUTINES::MESHS_INITIALISE(), CMISS_

MPI::MPI_ERROR_CHECK(), NODE_ROUTINES::NODE_CHECK_EXISTS(), NODE_ROUTINES::NODE_DESTROY(), NODE_ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_ROUTINES::NODE_NUMBER_SET(), NODE_ROUTINES::NODES_CREATE_FINISH(), NODE_ROUTINES::NODES_CREATE_START(), NODE_ROUTINES::NODES_FINALISE(), NODE_ROUTINES::NODES_INITIALISE(), MATHS::NORMALISE_DP(), MATHS::NORMALISE_SP(), STRINGS::NUMBER_TO_CHARACTER_DP(), STRINGS::NUMBER_TO_CHARACTER_INTG(), STRINGS::NUMBER_TO_CHARACTER_LINTG(), STRINGS::NUMBER_TO_CHARACTER_SP(), STRINGS::NUMBER_TO_VSTRING_SP(), BINARY_FILE::OPEN_BINARY_FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), CMISS_PARMETIS::PARMETIS_PARTKWAY(), CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF(), CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON(), CMISS_PETSC::PETSC_FINALIZE(), CMISS_PETSC::PETSC_INITIALIZE(), CMISS_PETSC::PETSC_ISDESTROY(), CMISS_PETSC::PETSC_ISFINALISE(), CMISS_PETSC::PETSC_ISINITIALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGDESTROY(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGFINALISE(), CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE(), CMISS_PETSC::PETSC_KSPCREATE(), CMISS_PETSC::PETSC_KSPDESTROY(), CMISS_PETSC::PETSC_KSPFINALISE(), CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETRESIDUALNORM(), CMISS_PETSC::PETSC_KSPINITIALISE(), CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(), CMISS_PETSC::PETSC_KSPSETOPERATORS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(), CMISS_PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PETSC_KSPSETUP(), CMISS_PETSC::PETSC_KSPSOLVE(), CMISS_PETSC::PETSC_LOGPRINTSUMMARY(), CMISS_PETSC::PETSC_MATASSEMBLYBEGIN(), CMISS_PETSC::PETSC_MATASSEMBLYEND(), CMISS_PETSC::PETSC_MATCREATE(), CMISS_PETSC::PETSC_MATCREATEMPIAIJ(), CMISS_PETSC::PETSC_MATCREATETEMPIDENSE(), CMISS_PETSC::PETSC_MATCREATESEQAIJ(), CMISS_PETSC::PETSC_MATCREATESEQDENSE(), CMISS_PETSC::PETSC_MATDESTROY(), CMISS_PETSC::PETSC_MATFINALISE(), CMISS_PETSC::PETSC_MATGETARRAY(), CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_MATINITIALISE(), CMISS_PETSC::PETSC_MATRESTOREARRAY(), CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING(), CMISS_PETSC::PETSC_MATSETOPTION(), CMISS_PETSC::PETSC_MATSETSIZES(), CMISS_PETSC::PETSC_MATSETVALUE(), CMISS_PETSC::PETSC_MATSETVALUELOCAL(), CMISS_PETSC::PETSC_MATSETVALUES(), CMISS_PETSC::PETSC_MATSETVALUESLOCAL(), CMISS_PETSC::PETSC_MATVIEW(), CMISS_PETSC::PETSC_MATZEROENTRIES(), CMISS_PETSC::PETSC_PCFINALISE(), CMISS_PETSC::PETSC_PCINITIALISE(), CMISS_PETSC::PETSC_PCSETTYPE(), CMISS_PETSC::PETSC_VECASSEMBLYBEGIN(), CMISS_PETSC::PETSC_VECASSEMBLYEND(), CMISS_PETSC::PETSC_VCCREATE(), CMISS_PETSC::PETSC_VCCREATEGHOST(), CMISS_PETSC::PETSC_VCCREATEGHOSTWITHARRAY(), CMISS_PETSC::PETSC_VCCREATEMPI(), CMISS_PETSC::PETSC_VCCREATEMPIWITHARRAY(), CMISS_PETSC::PETSC_VCCREATESEQ(), CMISS_PETSC::PETSC_VCCREATESEQWITHARRAY(), CMISS_PETSC::PETSC_VECDESTROY(), CMISS_PETSC::PETSC_VECDUPLICATE(), CMISS_PETSC::PETSC_VECFINALISE(), CMISS_PETSC::PETSC_VECGETARRAY(), CMISS_PETSC::PETSC_VECGETARRAYF90(), CMISS_PETSC::PETSC_VECGETLOCALSIZE(), CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE(), CMISS_PETSC::PETSC_VECGETSIZE(), CMISS_PETSC::PETSC_VECGETVALUES(), CMISS_PETSC::PETSC_VECHOSTGETLOCALFORM(), CMISS_PETSC::PETSC_VECHOSTSTORELOCALFORM(), CMISS_PETSC::PETSC_VECHOSTUPDATEBEGIN(), CMISS_PETSC::PETSC_VECHOSTUPDATEEND(), CMISS_PETSC::PETSC_VECINITIALISE(), CMISS_PETSC::PETSC_VECRESTOREARRAY(),

```

CMISS_PETSC::PETSC_VECRESTOREARRAYF90(),           CMISS_PETSC::PETSC_-
VECSET(),   CMISS_PETSC::PETSC_VECSETFROMOPTIONS(),   CMISS_PETSC::PETSC_-
VECSETLOCALTOGLOBALMAPPING(),   CMISS_PETSC::PETSC_VECSETSIZES(),   CMISS_-
PETSC::PETSC_VECSETVALUES(),      CMISS_PETSC::PETSC_VECSETVALUESLOCAL(), 
CMISS_PETSC::PETSC_VECVIEW(),    PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_-_
FINISH(),   PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START(),   PROBLEM_-
ROUTINES::PROBLEM_CONTROL_DESTROY(),      PROBLEM_ROUTINES::PROBLEM_-
CONTROL_FINALISE(),      PROBLEM_ROUTINES::PROBLEM_CONTROL_INITIALISE(), 
PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH(),   PROBLEM_ROUTINES::PROBLEM_-
CREATE_START(),   PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER(),   PROBLEM_-
ROUTINES::PROBLEM_DESTROY_PTR(),      PROBLEM_ROUTINES::PROBLEM_FINALISE(), 
PROBLEM_ROUTINES::PROBLEM_INITIALISE(),  PROBLEM_ROUTINES::PROBLEM_SETUP(), 
PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_ADD(),   PROBLEM_-
ROUTINES::PROBLEM_SOLUTION_FINALISE(),      PROBLEM_ROUTINES::PROBLEM_-
SOLUTION_INITIALISE(),      PROBLEM_ROUTINES::PROBLEM_SOLUTION_solve(), 
PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_FINISH(),      PROBLEM_-
ROUTINES::PROBLEM_SOLUTIONS_CREATE_START(),   PROBLEM_ROUTINES::PROBLEM_-
SOLUTIONS_FINALISE(),      PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE(), 
PROBLEM_ROUTINES::PROBLEM_SOLVE(),      PROBLEM_ROUTINES::PROBLEM_-
SOLVER_CREATE_FINISH(),      PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_-_
START(),   PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY(),      PROBLEM_-
ROUTINES::PROBLEM_SOLVER_GET(),  PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_-_
SET_NUMBER(), PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR(), PROBLEM_-
ROUTINES::PROBLEM_USER_NUMBER_FIND(),      PROBLEM_ROUTINES::PROBLEMS_-
FINALISE(),   PROBLEM_ROUTINES::PROBLEMS_INITIALISE(),   BINARY_FILE::READ_-_
BINARY_FILE_CHARACTER(),      BINARY_FILE::READ_BINARY_FILE_DP(),   BINARY_-_
FILE::READ_BINARY_FILE_DP1(),  BINARY_FILE::READ_BINARY_FILE_DPC(),   BINARY_-_
FILE::READ_BINARY_FILE_DPC1(),      BINARY_FILE::READ_BINARY_FILE_INTG(), 
BINARY_FILE::READ_BINARY_FILE_INTG1(),      BINARY_FILE::READ_BINARY_FILE_-_
LINTG(),      BINARY_FILE::READ_BINARY_FILE_LINTG1(),      BINARY_FILE::READ_-_
BINARY_FILE_LOGICAL(),      BINARY_FILE::READ_BINARY_FILE_LOGICAL1(),   BINARY_-_
FILE::READ_BINARY_FILE_SINTG(),      BINARY_FILE::READ_BINARY_FILE_SINTG1(), 
BINARY_FILE::READ_BINARY_FILE_SP(),      BINARY_FILE::READ_BINARY_FILE_SP1(), 
BINARY_FILE::READ_BINARY_FILE_SPC(),      BINARY_FILE::READ_BINARY_FILE_-_
SPC1(),      BINARY_FILE::READ_BINARY_TAG_HEADER(),   REGION_ROUTINES::REGION_-_
COORDINATE_SYSTEM_GET(),      REGION_ROUTINES::REGION_COORDINATE_SYSTEM_-_
SET_NUMBER(),      REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR(), 
REGION_ROUTINES::REGION_CREATE_FINISH(),   REGION_ROUTINES::REGION_CREATE_-_
START(),   REGION_ROUTINES::REGION_DESTROY(),      REGION_ROUTINES::REGION_-_
LABEL_GET(),      REGION_ROUTINES::REGION_LABEL_SET_NUMBER(),   REGION_-_
ROUTINES::REGION_LABEL_SET_PTR(),      REGION_ROUTINES::REGION_SUB_REGION_-_
CREATE_FINISH(),      REGION_ROUTINES::REGION_SUB_REGION_CREATE_START(), 
REGION_ROUTINES::REGION_USER_NUMBER_FIND(),      REGION_ROUTINES::REGION_-_
USER_NUMBER_FIND_PTR(),      REGION_ROUTINES::REGIONS_FINALISE(),   REGION_-_
ROUTINES::REGIONS_INITIALISE(),      BINARY_FILE::RESET_BINARY_NUMBER_TAGS(), 
BINARY_FILE::SET_BINARY_FILE(),      SORTING::SHELL_SORT_DP(),   SORTING::SHELL_-_
SORT_INTG(),  SORTING::SHELL_SORT_SP(), BINARY_FILE::SKIP_BINARY_FILE(), BINARY_-_
FILE::SKIP_BINARY_TAGS(), BINARY_FILE::SKIP_CM_BINARY_HEADER(), MATHS::SOLVE_-_
SMALL_LINEAR_SYSTEM_DP(), MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP(), SOLVER_-_
ROUTINES::SOLVER_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_CREATE_-_
START(),      SOLVER_ROUTINES::SOLVER_DESTROY(),      SOLVER_ROUTINES::SOLVER_-_
EIGENPROBLEM_CREATE_FINISH(),      SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_-_
FINALISE(),      SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE(),   SOLVER_-_
ROUTINES::SOLVER_EIGENPROBLEM_SOLVE(),  SOLVER_ROUTINES::SOLVER_FINALISE(),

```

SOLVER_ROUTINES::SOLVER_INITIALISE(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(),
 SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_-
 LINEAR_DIRECT_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_-
 FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE(), SOLVER_ROUTINES::SOLVER_LINEAR_-
 DIRECT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_-
 ITERATIVE_ABSOLUTE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_-
 ITERATIVE_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_-
 DIVERGENCE_TOLERANCE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_-
 FINALISE(), SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET(), SOLVER_-
 ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 LINEAR_ITERATIVE_TYPE_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE(),
 SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET(), SOLVER_ROUTINES::SOLVER_-
 MATRICES_ASSEMBLE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_-
 FINISH(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(),
 SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY(), SOLVER_MATRICES_-
 ROUTINES::SOLVER_MATRICES_FINALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_-
 MATRICES_INITIALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_-
 TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT(), SOLVER_-
 MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(), SOLVER_MATRICES_-
 ROUTINES::SOLVER_MATRIX_FINALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_-
 MATRIX_INITIALISE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_-
 CALCULATE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH(), SOLVER_-
 ROUTINES::SOLVER_NONLINEAR_FINALISE(), SOLVER_ROUTINES::SOLVER_-
 NONLINEAR_INITIALISE(), SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE(), SOLVER_-
 ROUTINES::SOLVER_OUTPUT_TYPE_SET(), SOLVER_ROUTINES::SOLVER_SOLVE(),
 SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET(), SOLVER_ROUTINES::SOLVER_TIME_-
 INTEGRATION_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_-
 FINALISE(), SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE(), SOLVER_-
 ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE(), SOLVER_ROUTINES::SOLVER_-
 VARIABLES_UPDATE(), STRINGS::STRING_TO_DOUBLE_C(), STRINGS::STRING_-
 TO_DOUBLE_VS(), STRINGS::STRING_TO_INTEGER_C(), STRINGS::STRING_TO_-
 INTEGER_VS(), STRINGS::STRING_TO_LOGICAL_C(), STRINGS::STRING_TO_LOGICAL_-
 VS(), STRINGS::STRING_TO_LONG_INTEGER_C(), STRINGS::STRING_TO_LONG_-
 INTEGER_VS(), FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS(), FIELD_-
 IO_ROUTINES::STRING_TO_MUTI_REALS_VS(), STRINGS::STRING_TO_SINGLE_C(),
 STRINGS::STRING_TO_SINGLE_VS(), TREES::TREE_CREATE_FINISH(), TREES::TREE_-
 CREATE_START(), TREES::TREE_DESTROY(), TREES::TREE_DETACH_AND_DESTROY(),
 TREES::TREE_DETACH_IN_ORDER(), TREES::TREE_FINALISE(), TREES::TREE_INITIALISE(),
 TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_DELETE(), TREES::TREE_-
 ITEM_INSERT(), TREES::TREE_NODE_FINALISE(), TREES::TREE_NODE_INITIALISE(),
 TREES::TREE_NODE_KEY_GET(), TREES::TREE_NODE_VALUE_GET(), TREES::TREE_-
 NODE_VALUE_SET(), TREES::TREE_OUTPUT(), TREES::TREE_OUTPUT_IN_ORDER(),
 TREES::TREE_PREDECESSOR(), TREES::TREE_SEARCH(), TREES::TREE_SUCCESSOR(),
 MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L(), MATRIX_-
 VECTOR::VECTOR_ALL_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_CREATE_FINISH(),
 MATRIX_VECTOR::VECTOR_CREATE_START(), MATRIX_VECTOR::VECTOR_DATA_GET_-
 DP(), MATRIX_VECTOR::VECTOR_DATA_GET_INTG(), MATRIX_VECTOR::VECTOR_DATA_-
 GET_L(), MATRIX_VECTOR::VECTOR_DATA_GET_SP(), MATRIX_VECTOR::VECTOR_DATA_-

TYPE_SET(), MATRIX_VECTOR::VECTOR_DESTROY(), MATRIX_VECTOR::VECTOR_FINALISE(), MATRIX_VECTOR::VECTOR_INITIALISE(), MATRIX_VECTOR::VECTOR_SIZE_SET(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP(), MATRIX_VECTOR::VECTOR_VALUES_GET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_GET_L0(), MATRIX_VECTOR::VECTOR_VALUES_GET_L10(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP(), MATRIX_VECTOR::VECTOR_VALUES_GET_SP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP(), MATRIX_VECTOR::VECTOR_VALUES_SET_DP1(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG(), MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1(), MATRIX_VECTOR::VECTOR_VALUES_SET_L0(), MATRIX_VECTOR::VECTOR_VALUES_SET_L10(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP(), MATRIX_VECTOR::VECTOR_VALUES_SET_SP1(), BINARY_FILE::WRITE_BINARY_FILE_CHARACTER(), BINARY_FILE::WRITE_BINARY_FILE_DP(), BINARY_FILE::WRITE_BINARY_FILE_DP1(), BINARY_FILE::WRITE_BINARY_FILE_DPC(), BINARY_FILE::WRITE_BINARY_FILE_DPC1(), BINARY_FILE::WRITE_BINARY_FILE_INTG(), BINARY_FILE::WRITE_BINARY_FILE_INTG1(), BINARY_FILE::WRITE_BINARY_FILE_LINTG(), BINARY_FILE::WRITE_BINARY_FILE_LINTG1(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL(), BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1(), BINARY_FILE::WRITE_BINARY_FILE_SINTG(), BINARY_FILE::WRITE_BINARY_FILE_SINTG1(), BINARY_FILE::WRITE_BINARY_FILE_SP(), BINARY_FILE::WRITE_BINARY_FILE_SP1(), BINARY_FILE::WRITE_BINARY_FILE_SPC(), BINARY_FILE::WRITE_BINARY_FILE_SPC1(), and BINARY_FILE::WRITE_BINARY_TAG_HEADER().

6.1.2.8 subroutine **BASE_ROUTINES::FLAG_ERROR_C** (**CHARACTER(LEN=*)**,**intent(in)** **STRING**, **INTEGER(INTG)**,**intent(out)** **ERR**, **TYPE(VARYING_STRING)**,**intent(out)** **ERROR**, ***/** [private]

Sets the error string specified by a character string and flags an error.

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 470 of file base_routines.f90.

6.1.2.9 subroutine **BASE_ROUTINES::FLAG_ERROR_VS** (**TYPE(VARYING_STRING)**,**intent(in)** **STRING**, **INTEGER(INTG)**,**intent(out)** **ERR**, **TYPE(VARYING_STRING)**,**intent(out)** **ERROR**, ***/** [private]

Sets the error string specified by a varying string and flags an error.

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 491 of file base_routines.f90.

6.1.2.10 subroutine BASE_ROUTINES::FLAG_WARNING_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Writes a warning message specified by a character string to the user.

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 510 of file base_routines.f90.

6.1.2.11 subroutine BASE_ROUTINES::FLAG_WARNING_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Writes a warning message specified by a varying string to the user.

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 531 of file base_routines.f90.

6.1.2.12 subroutine BASE_ROUTINES::OUTPUT_SET_OFF (INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets writes file echo output off.

Parameters:

ERR The error code

ERROR The error string

Definition at line 830 of file base_routines.f90.

6.1.2.13 subroutine BASE_ROUTINES::OUTPUT_SET_ON (CHARACTER(LEN=*),intent(in) ECHO_FILENAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets writes file echo output on.

Parameters:

ECHO_FILENAME The filename of the file to echo output to

ERR The error code

ERROR The error string

Definition at line 858 of file base_routines.f90.

6.1.2.14 subroutine `BASE_ROUTINES::TIMING_SET_OFF` (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets timing off.

Parameters:

ERR The error code

ERROR The error string

Definition at line 890 of file base_routines.f90.

6.1.2.15 subroutine `BASE_ROUTINES::TIMING_SET_ON` (INTEGER(INTG),intent(in) *TIMING_TYPE*, LOGICAL,intent(in) *TIMING_SUMMARY_FLAG*, CHARACTER(LEN=*),intent(in) *TIMING_FILENAME*, CHARACTER(LEN=*),dimension(:,),intent(in) *ROUTINE_LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets timing on.

Parameters:

TIMING_TYPE The type of timing to set on

See also:

[BASE_ROUTINES::TimingTypes](#)

TIMING_SUMMARY_FLAG .TRUE. if the timing information will be output with subsequent *TIMING_SUMMARY_OUTPUT* calls, .FALSE. if the timing information will be output every time the routine exits

TIMING_FILENAME If present the name of the file to output timing information to. If omitted the timing output is sent to the screen

ROUTINE_LIST The list of routines to set timing on in.

ERR The error code

ERROR The error string

Definition at line 936 of file base_routines.f90.

References KINDS::_SP.

6.1.2.16 subroutine `BASE_ROUTINES::TIMING_SUMMARY_OUTPUT` (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Outputs the timing summary.

Parameters:

ERR The error code

ERROR The error string

Definition at line 1059 of file base_routines.f90.

6.1.2.17 subroutine BASE_ROUTINES::WRITE_STR (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Writes the output string to a specified output stream.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

ERR The error code

ERROR The error string

Definition at line 1121 of file base_routines.f90.

References MACHINE_CONSTANTS::MACHINE_OS.

Referenced by INPUT_OUTPUT::WR(), INPUT_OUTPUT::WRITE_STRING_C(), INPUT_OUTPUT::WRITE_STRING_FMT(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS(), INPUT_OUTPUT::WRITE_STRING_IDX(), INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(), INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_L(), INPUT_OUTPUT::WRITE_STRING_MATRIX_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(), INPUT_OUTPUT::WRITE_STRING_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_VALUE_VS(), and INPUT_OUTPUT::WRITE_STRING_VS().

6.1.3 Variable Documentation

6.1.3.1 LOGICAL,save BASE_ROUTINES::DIAG_ALL_SUBROUTINES

.TRUE. if diagnostic output is required in all routines

Definition at line 156 of file base_routines.f90.

6.1.3.2 LOGICAL,save BASE_ROUTINES::DIAG_FILE_OPEN

.TRUE. if the diagnostic output file is open

Definition at line 158 of file base_routines.f90.

6.1.3.3 LOGICAL,save BASE_ROUTINES::DIAG_FROM_SUBROUTINE

.TRUE. if diagnostic output is required from a particular routine

Definition at line 157 of file base_routines.f90.

6.1.3.4 LOGICAL,save BASE_ROUTINES::DIAG_OR_TIMING

.TRUE. if diagnostics or time is .TRUE.

Definition at line 159 of file base_routines.f90.

6.1.3.5 TYPE(ROUTINE_LIST_TYPE),save BASE_ROUTINES::DIAG_ROUTINE_LIST

The list of routines for which diagnostic output is required.

Definition at line 167 of file base_routines.f90.

6.1.3.6 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS

.TRUE. if diagnostic output is required in any routines.

Definition at line 145 of file base_routines.f90.

6.1.3.7 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS1

.TRUE. if level 1 diagnostic output is active in the current routine

Definition at line 146 of file base_routines.f90.

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(),

`FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `MESH_ROUTINES::MESH_CREATE_FINISH()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`, `NODE_ROUTINES::NODES_CREATE_FINISH()`, `PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH()`, `REGION_ROUTINES::REGION_CREATE_FINISH()`, `REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

6.1.3.8 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS2

.TRUE. if level 2 diagnostic output is active in the current routine

Definition at line 147 of file base_routines.f90.

Referenced by `COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE()`, `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, and `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

6.1.3.9 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS3

.TRUE. if level 3 diagnostic output is active in the current routine

Definition at line 148 of file base_routines.f90.

Referenced by `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE()`.

6.1.3.10 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS4

.TRUE. if level 4 diagnostic output is active in the current routine

Definition at line 149 of file base_routines.f90.

6.1.3.11 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS5

.TRUE. if level 5 diagnostic output is active in the current routine

Definition at line 150 of file base_routines.f90.

6.1.3.12 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL1

.TRUE. if the user has requested level 1 diagnostic output to be active

Definition at line 151 of file base_routines.f90.

6.1.3.13 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL2

.TRUE. if the user has requested level 2 diagnostic output to be active

Definition at line 152 of file base_routines.f90.

6.1.3.14 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL3

.TRUE. if the user has requested level 3 diagnostic output to be active

Definition at line 153 of file base_routines.f90.

6.1.3.15 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL4

.TRUE. if the user has requested level 4 diagnostic output to be active

Definition at line 154 of file base_routines.f90.

6.1.3.16 LOGICAL,save BASE_ROUTINES::DIAGNOSTICS_LEVEL5

.TRUE. if the user has requested level 5 diagnostic output to be active

Definition at line 155 of file base_routines.f90.

6.1.3.17 LOGICAL,save BASE_ROUTINES::ECHO_OUTPUT

.TRUE. if all output is to be echoed to the echo file

Definition at line 160 of file base_routines.f90.

6.1.3.18 INTEGER(INTG),parameter BASE_ROUTINES::MAX_OUTPUT_LINES = 500

Maximum number of lines that can be output.

See also:

[BASE_ROUTINES::WRITE_STR](#)

Definition at line 57 of file base_routines.f90.

6.1.3.19 CHARACTER(LEN=MAXSTRLEN),save BASE_ROUTINES::OP_STRING

The array of lines to output.

Definition at line 166 of file base_routines.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE(), INPUT_OUTPUT::WR(), INPUT_OUTPUT::WRITE_STRING_C(), INPUT_OUTPUT::WRITE_STRING_FMT(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS(), INPUT_OUTPUT::WRITE_STRING_IDX(), INPUT_OUTPUT::WRITE_STRING_MATRIX_DP(), INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_L(), INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG(), INPUT_OUTPUT::WRITE_STRING_MATRIX_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L(), INPUT_OUTPUT::WRITE_STRING_

TWO_VALUE_C_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS(), INPUT_OUTPUT::WRITE_STRING_VALUE_C(), INPUT_OUTPUT::WRITE_STRING_VALUE_DP(), INPUT_OUTPUT::WRITE_STRING_VALUE_INTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG(), INPUT_OUTPUT::WRITE_STRING_VALUE_SP(), INPUT_OUTPUT::WRITE_STRING_VALUE_VS(), and INPUT_OUTPUT::WRITE_STRING_VS().

6.1.3.20 TYPE(ROUTINE_STACK_TYPE),save BASE_ROUTINES::ROUTINE_STACK

The routime invocation stack.

Definition at line 169 of file base_routines.f90.

6.1.3.21 LOGICAL,save BASE_ROUTINES::TIMING

.TRUE. if timing output is required in any routines.

Definition at line 161 of file base_routines.f90.

6.1.3.22 LOGICAL,save BASE_ROUTINES::TIMING_ALL_SUBROUTINES

.TRUE. if timing output is required in all routines

Definition at line 163 of file base_routines.f90.

6.1.3.23 LOGICAL,save BASE_ROUTINES::TIMING_FILE_OPEN

.TRUE. if the timing output file is open

Definition at line 165 of file base_routines.f90.

6.1.3.24 LOGICAL,save BASE_ROUTINES::TIMING_FROM_SUBROUTINE

.TRUE. if timing output is required from a particular routine

Definition at line 164 of file base_routines.f90.

6.1.3.25 TYPE(ROUTINE_LIST_TYPE),save BASE_ROUTINES::TIMING_ROUTINE_LIST

The list of routines for which timing output is required.

Definition at line 168 of file base_routines.f90.

6.1.3.26 LOGICAL,save BASE_ROUTINES::TIMING_SUMMARY

.TRUE. if timing output will be summary form via a TIMING_SUMMARY_OUTPUT call otherwise timing will be output for routines when the routine exits

See also:

[BASE_ROUTINES::TIMING_SUMMARY_OUTPUT](#)

Definition at line 162 of file base_routines.f90.

6.2 BINARY_FILE Namespace Reference

This module handles the reading and writing of binary files.

Classes

- struct [BINARY_FILE_INFO_TYPE](#)
- struct [BINARY_FILE_TYPE](#)
- struct [BINARY_TAG_TYPE](#)
- interface [interface](#)
- interface [READ_BINARY_FILE](#)
- interface [WRITE_BINARY_FILE](#)

Functions

- LOGICAL [INQUIRE_OPEN_BINARY_FILE](#) (FILEID)
- LOGICAL [INQUIRE_EOF_BINARY_FILE](#) (FILEID, ERR, ERROR)
- subroutine [CLOSE_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [CLOSE_CMISS_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [OPEN_BINARY_FILE](#) (FILEID, COMMAND, FILENAME, ERR, ERROR,*)
- subroutine [OPEN_CMISS_BINARY_FILE](#) (FILEID, FILE_TYPE, NUMBER_TA S,&VERSION, FILEVERSION, COMMAND, EXTENSION, FILENAME, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_CHARACTER](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_TAG_HEADER](#) (FILEID, TAG, ERR, ERROR,*)
- subroutine [RESET_BINARY_NUMBER_TAGS](#) (FILEID, NUMBER_TAGS, ERR, ERROR,*)
- subroutine [SET_BINARY_FILE](#) (FILEID, SET_CODE, ERR, ERROR,*)
- subroutine [SKIP_CM_BINARY_HEADER](#) (FILEID, SKIP, ERR, ERROR,*)
- subroutine [SKIP_BINARY_FILE](#) (FILEID, NUMBER_BYTES, ERR, ERROR,*)
- subroutine [SKIP_BINARY_TAGS](#) (FILEID, TAG, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [WRITE_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

- subroutine `WRITE_BINARY_FILE_SINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_TAG_HEADER` (FILEID, TAG, ERR, ERROR,*)

Variables

- INTEGER(INTG), dimension, parameter `MAX_NUM_BINARY_FILES` = 99
- INTEGER(INTG), parameter `FILE_BEGINNING` = 0
- INTEGER(INTG), parameter `FILE_CURRENT` = 1
- INTEGER(INTG), parameter `FILE_END` = 2
- INTEGER(INTG), parameter `FILE SAME_ENDIAN` = 0
- INTEGER(INTG), parameter `FILE_CHANGE_ENDIAN` = 1
- INTEGER(INTG), parameter `BINARY_FILE_READABLE` = 1
- INTEGER(INTG), parameter `BINARY_FILE_WRITABLE` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_IDENTITY` = 7
- INTEGER(INTG), parameter `CMISS_BINARY_MATRIX_FILE` = 1
- INTEGER(INTG), parameter `CMISS_BINARY_HISTORY_FILE` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_SIGNAL_FILE` = 3
- INTEGER(INTG), parameter `CMISS_BINARY_IDENTITY_HEADER` = 1
- INTEGER(INTG), parameter `CMISS_BINARY_MACHINE_HEADER` = 2
- INTEGER(INTG), parameter `CMISS_BINARY_FILE_HEADER` = 3
- LOGICAL, save `BINARY_FILE_USED` = .FALSE.

6.2.1 Detailed Description

This module handles the reading and writing of binary files.

6.2.2 Function Documentation

6.2.2.1 subroutine `BINARY_FILE::CLOSE_BINARY_FILE` (TYPE(BINARY_- FILE_TYPE),intent(out) `FILEID`, INTEGER(INTG),intent(out) `ERR`, TYPE(VARYING_STRING),intent(out) `ERROR`, *)

Definition at line 474 of file `binary_file.f90`.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.2 subroutine BINARY_FILE::CLOSE_CMISS_BINARY_FILE (TYPE(BINARY_-
FILE_TYPE),intent(inout) FILEID, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 518 of file binary_file_f.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.3 LOGICAL BINARY_FILE::INQUIRE_EOF_BINARY_FILE (TYPE(BINARY_-
FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR)**

Definition at line 430 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.4 LOGICAL BINARY_FILE::INQUIRE_OPEN_BINARY_FILE
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID)**

Definition at line 407 of file binary_file_f.f90.

**6.2.2.5 subroutine BINARY_FILE::OPEN_BINARY_FILE (TYPE(BINARY_FILE_-
TYPE),intent(out) FILEID, CHARACTER(LEN=*),intent(in) COMMAND,
CHARACTER(LEN=*),intent(in) FILENAME, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 547 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), F90C::F2CSTRING(), and MACHINE_CONSTANTS::MACHINE_-ENDIAN.

Here is the call graph for this function:

**6.2.2.6 subroutine BINARY_FILE::OPEN_CMISS_BINARY_FILE (TYPE(BINARY_FILE_-
TYPE),intent(out) FILEID, INTEGER(INTG),intent(in) FILE_TYPE, NUMBER_TA
S, &,dimension(3),intent(in) VERSION, INTEGER(INTG),dimension(3),intent(out)
FILEVERSION, CHARACTER(LEN=*),intent(in) COMMAND,
CHARACTER(LEN=*),intent(in) EXTENSION, CHARACTER(LEN=*),intent(in)
FILENAME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Definition at line 623 of file binary_file_f.f90.

References MACHINE_CONSTANTS::CHARACTER_SIZE, MACHINE_CONSTANTS::DOUBLE_-COMPLEX_SIZE, MACHINE_CONSTANTS::DOUBLE_REAL_SIZE, BASE_-ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), BASE_-ROUTINES::GENERAL_OUTPUT_TYPE, MACHINE_CONSTANTS::INTEGER_SIZE,

KINDS::INTG, MACHINE_CONSTANTS::LOGICAL_SIZE, MACHINE_CONSTANTS::LONG_INTEGER_SIZE, MACHINE_CONSTANTS::MACHINE_CHAR_FORMAT, MACHINE_CONSTANTS::MACHINE_DP_FORMAT, MACHINE_CONSTANTS::MACHINE_ENDIAN, MACHINE_CONSTANTS::MACHINE_INT_FORMAT, MACHINE_CONSTANTS::MACHINE_OS, MACHINE_CONSTANTS::MACHINE_SP_FORMAT, MACHINE_CONSTANTS::MACHINE_TYPE, BASE_ROUTINES::OP_STRING, MACHINE_CONSTANTS::SHORT_INTEGER_SIZE, MACHINE_CONSTANTS::SINGLE_COMPLEX_SIZE, and MACHINE_CONSTANTS::SINGLE_REAL_SIZE.

Here is the call graph for this function:

6.2.2.7 subroutine BINARY_FILE::READ_BINARY_FILE_CHARACTER
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, CHARACTER(LEN=*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1341 of file binary_file_f.f90.

References F90C::C2FSTRING(), F90C::CSTRINGLENGTH(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.8 subroutine BINARY_FILE::READ_BINARY_FILE_DP (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(DP),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1243 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.9 subroutine BINARY_FILE::READ_BINARY_FILE_DP1 (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(DP),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1290 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.10 subroutine BINARY_FILE::READ_BINARY_FILE_DPC (TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(DPC),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1591 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.11 subroutine `BINARY_FILE::READ_BINARY_FILE_DPC1` (TYPE(BINARY_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*, COMPLEX(DPC),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 1639 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.12 subroutine `BINARY_FILE::READ_BINARY_FILE_INTG` (TYPE(BINARY_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*, INTEGER(INTG),dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 851 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.13 subroutine `BINARY_FILE::READ_BINARY_FILE_INTG1` (TYPE(BINARY_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*, INTEGER(INTG),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 898 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.14 subroutine `BINARY_FILE::READ_BINARY_FILE_LINTG` (TYPE(BINARY_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*, INTEGER(LINTG),dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 1047 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.15 subroutine `BINARY_FILE::READ_BINARY_FILE_LINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(LINTG),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1094 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.16 subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1393 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.17 subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL1`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1440 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.18 subroutine `BINARY_FILE::READ_BINARY_FILE_SINTG` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 949 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.19 subroutine `BINARY_FILE::READ_BINARY_FILE_SINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 996 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.20 subroutine BINARY_FILE::READ_BINARY_FILE_SP (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1145 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.21 subroutine BINARY_FILE::READ_BINARY_FILE_SP1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1192 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.22 subroutine BINARY_FILE::READ_BINARY_FILE_SPC (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1491 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

6.2.2.23 subroutine BINARY_FILE::READ_BINARY_FILE_SPC1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 1539 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MACHINE_CONSTANTS::MACHINE_ENDIAN.

Here is the call graph for this function:

**6.2.2.24 subroutine `BINARY_FILE::READ_BINARY_TAG_HEADER` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, TYPE(BINARY_TAG_TYPE),intent(out) *TAG*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1691 of file binary_file_f.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.2.2.25 subroutine `BINARY_FILE::RESET_BINARY_NUMBER_TAGS` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUMBER_TAGS*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1749 of file binary_file_f.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, `MACHINE_CONSTANTS::INTEGER_SIZE`, and `MACHINE_-CONSTANTS::SINGLE_REAL_SIZE`.

Here is the call graph for this function:

**6.2.2.26 subroutine `BINARY_FILE::SET_BINARY_FILE` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *SET_CODE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1803 of file binary_file_f.f90.

References `F90C::C2FSTRING()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.2.2.27 subroutine `BINARY_FILE::SKIP_BINARY_FILE` (TYPE(BINARY_FILE_-
TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUMBER_BYTES*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1918 of file binary_file_f.f90.

References `F90C::C2FSTRING()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.2.2.28 subroutine `BINARY_FILE::SKIP_BINARY_TAGS` (TYPE(BINARY_FILE_-
TYPE),intent(in) *FILEID*, TYPE(BINARY_TAG_TYPE),intent(in) *TAG*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1960 of file binary_file_f.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.2.2.29 subroutine `BINARY_FILE::SKIP_CM_BINARY_HEADER` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *SKIP*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 1846 of file binary_file.f90.

References MACHINE_CONSTANTS::CHARACTER_SIZE, BASE_ROUTINES::ENTERS(), BASE_-ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MACHINE_CONSTANTS::INTEGER_SIZE, and MACHINE_CONSTANTS::SINGLE_REAL_SIZE.

Here is the call graph for this function:

6.2.2.30 subroutine `BINARY_FILE::WRITE_BINARY_FILE_CHARACTER` (TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*, CHARACTER(LEN=*),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 2466 of file binary_file.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and F90C::F2cstring().

Here is the call graph for this function:

**6.2.2.31 subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
REAL(DP),dimension(*),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2376 of file binary_file.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.32 subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
REAL(DP),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2420 of file binary_file.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.33 subroutine `BINARY_FILE::WRITE_BINARY_FILE_DPC` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
COMPLEX(DPC),dimension(*),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2689 of file binary_file.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.34 subroutine BINARY_FILE::WRITE_BINARY_FILE_DPC1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(DPC),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2733 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.35 subroutine BINARY_FILE::WRITE_BINARY_FILE_INTG (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(INTG),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2022 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.36 subroutine BINARY_FILE::WRITE_BINARY_FILE_INTG1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(INTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2063 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.37 subroutine BINARY_FILE::WRITE_BINARY_FILE_LINTG (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, INTEGER(LINTG),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2197 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.38 subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(LINTG),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2240 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.39 subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,dimension(*),intent(in) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2510 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.40 subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1`
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, LOGICAL,intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2552 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.41 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),dimension(*),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2108 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.2.2.42 subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG1` (TYPE(BINARY_-
FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in) *NUM_DATA*,
INTEGER(SINTG),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 2151 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.43 subroutine BINARY_FILE::WRITE_BINARY_FILE_SP (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2286 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.44 subroutine BINARY_FILE::WRITE_BINARY_FILE_SP1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, REAL(SP),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2330 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.45 subroutine BINARY_FILE::WRITE_BINARY_FILE_SPC (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2598 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.46 subroutine BINARY_FILE::WRITE_BINARY_FILE_SPC1 (TYPE(BINARY_-FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_DATA, COMPLEX(SPC),intent(in) DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 2642 of file binary_file_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.2.2.47 subroutine `BINARY_FILE::WRITE_BINARY_TAG_HEADER` (`TYPE(BINARY_FILE_TYPE)`,`intent(in)` `FILEID`, `TYPE(BINARY_TAG_TYPE)`,`intent(out)` `TAG`, `INTEGER(INTG)`,`intent(out)` `ERR`, `TYPE(VARYING_STRING)`,`intent(out)` `ERROR`, `*`)

Definition at line 2780 of file binary_file_f.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.2.3 Variable Documentation

6.2.3.1 `INTEGER(INTG),parameter BINARY_FILE::BINARY_FILE_READABLE = 1`

Definition at line 249 of file binary_file_f.f90.

6.2.3.2 `LOGICAL,save BINARY_FILE::BINARY_FILE_USED = .FALSE.`

Definition at line 301 of file binary_file_f.f90.

6.2.3.3 `INTEGER(INTG),parameter BINARY_FILE::BINARY_FILE_WRITABLE = 2`

Definition at line 250 of file binary_file_f.f90.

6.2.3.4 `INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_FILE_HEADER = 3`

Definition at line 259 of file binary_file_f.f90.

6.2.3.5 `INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_HISTORY_FILE = 2`

Definition at line 255 of file binary_file_f.f90.

6.2.3.6 `INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_IDENTITY = 7`

Definition at line 253 of file binary_file_f.f90.

6.2.3.7 `INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_IDENTITY_HEADER = 1`

Definition at line 257 of file binary_file_f.f90.

6.2.3.8 `INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_MACHINE_HEADER = 2`

Definition at line 258 of file binary_file_f.f90.

6.2.3.9 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_MATRIX_FILE = 1

Definition at line 254 of file binary_file_f.f90.

6.2.3.10 INTEGER(INTG),parameter BINARY_FILE::CMISS_BINARY_SIGNAL_FILE = 3

Definition at line 256 of file binary_file_f.f90.

6.2.3.11 INTEGER(INTG),parameter BINARY_FILE::FILE_BEGINNING = 0

Definition at line 240 of file binary_file_f.f90.

6.2.3.12 INTEGER(INTG),parameter BINARY_FILE::FILE_CHANGE_ENDIAN = 1

Definition at line 246 of file binary_file_f.f90.

6.2.3.13 INTEGER(INTG),parameter BINARY_FILE::FILE_CURRENT = 1

Definition at line 241 of file binary_file_f.f90.

6.2.3.14 INTEGER(INTG),parameter BINARY_FILE::FILE_END = 2

Definition at line 242 of file binary_file_f.f90.

6.2.3.15 INTEGER(INTG),parameter BINARY_FILE::FILE_SAME_ENDIAN = 0

Definition at line 245 of file binary_file_f.f90.

6.2.3.16 INTEGER(INTG),dimension,parameter BINARY_FILE::MAX_NUM_BINARY_FILES = 99

Definition at line 237 of file binary_file_f.f90.

6.3 BLAS Namespace Reference

This module contains the [interface](#) descriptions to the [BLAS](#) routines.

Classes

- interface [interface](#)

6.3.1 Detailed Description

This module contains the [interface](#) descriptions to the [BLAS](#) routines.

6.4 CLASSICAL_FIELD_ROUTINES Namespace Reference

This module handles all classical field class routines.

Functions

- subroutine `CL` (`EQUATIONS_SET`, `EQUATIONS_TYPE`, `EQUATIONS_SUBTYPE`,`&ERR`, `ERROR,*`)
Sets/changes the problem type and subtype for a classical field equation set class.
- subroutine `CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE` (`EQUATIONS_SET`, `ELEMENT_NUMBER`, `ERR`, `ERROR,*`)
Calculates the element stiffness matrices and rhs vector for the given element number for a classical field class finite element equation set.
- subroutine `CLASSICAL_FIELD_EQUATIONS_SET_SETUP` (`EQUATIONS_SET`, `SETUP_TYPE`, `ACTION_TYPE`, `ERR`, `ERROR,*`)
Sets up the equations set for a classical field equations set class.
- subroutine `CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET` (`PROBLEM`, `PROBLEM_EQUATION_TYPE`, `PROBLEM_SUBTYPE`, `ERR`, `ERROR,*`)
Sets/changes the problem type and subtype for a classical field problem class.
- subroutine `CLASSICAL_FIELD_PROBLEM_SETUP` (`PROBLEM`, `SETUP_TYPE`, `ACTION_TYPE`, `ERR`, `ERROR,*`)
Sets up the problem for a classical field problem class.

6.4.1 Detailed Description

This module handles all classical field class routines.

6.4.2 Function Documentation

6.4.2.1 subroutine CLASSICAL_FIELD_ROUTINES::CL (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *EQUATIONS_TYPE*, INTEGER(INTG),intent(in) *EQUATIONS_SUBTYPE*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)

Sets/changes the problem type and subtype for a classical field equation set class.

Parameters:

- EQUATIONS_SET* A pointer to the equations set
EQUATIONS_TYPE The equation type
EQUATIONS_SUBTYPE The equation subtype

Definition at line 77 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-ADVECTION_DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-BIHARMONIC_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-HELMHOLTZ_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-LAPLACE_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-POISSON_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-REACTION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-WAVE_EQUATION_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET()`.

Here is the call graph for this function:

```
6.4.2.2 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_-
SETFINITEELEMENTCALCULATE (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(in) ELEMENT_NUMBER,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]
```

Calculates the element stiffness matrices and rhs vector for the given element number for a clasical field class finite element equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set

ELEMENT_NUMBER The element number to calcualate

ERR The error code

ERROR The error string

Definition at line 130 of file `classical_field_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-ADVECTION_DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-BIHARMONIC_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-DIFFUSION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-HELMHOLTZ_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-LAPLACE_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-POISSON_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-REACTION_EQUATION_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-WAVE_EQUATION_TYPE`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITEELEMENT_-CALCULATE()`.

Referenced by `EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITEELEMENT_-CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.4.2.3 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets up the equations set for a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 181 of file classical_field_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BIHAMONIC_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_REACTION_DIFFUSION_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_EQUATION_TYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP().

Here is the call graph for this function:

Here is the caller graph for this function:

6.4.2.4 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(in) *PROBLEM_EQUATION_TYPE*, INTEGER(INTG),intent(in) *PROBLEM_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the problem type and subtype for a classical field problem class.

Parameters:

PROBLEM A pointer to the problem

PROBLEM_EQUATION_TYPE The problem type

PROBLEM_SUBTYPE The problem subtype

ERR The error code

ERROR The error string

Definition at line 233 of file classical_field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET()`, `PROBLEM_CONSTANTS::PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_BIHAMONIC_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_HELMHOLTZ_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_WAVE_EQUATION_TYPE`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.4.2.5 subroutine CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_-
PROBLEM_SETUP (TYPE(PROBLEM_TYPE),pointer *PROBLEM*,
INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_TYPE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Sets up the problem for a classical field problem class.

Parameters:

PROBLEM A pointer to the problem

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 285 of file `classical_field_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_BIHAMONIC_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_HELMHOLTZ_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_WAVE_EQUATION_TYPE`.

Referenced by `PROBLEM_ROUTINES::PROBLEM_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.5 CMISS Namespace Reference

The top level cmiss module.

Functions

- subroutine [CMISS_FINALISE](#) (ERR, ERROR,*)

Finalises CMISS.
- subroutine [CMISS_INITIALISE](#) (ERR, ERROR,*)

Initialises CMISS.
- subroutine [CMISS_WRITE_ERROR](#) (ERR, ERROR)

Writes the error string to screen.

Variables

- INTEGER(INTG), parameter [CMISS_MAJOR_VERSION](#) = 0
- INTEGER(INTG), parameter [CMISS_MINOR_VERSION](#) = 1
- INTEGER(INTG), parameter [CMISS_BUILD_VERSION](#) = 1

6.5.1 Detailed Description

The top level cmiss module.

6.5.2 Function Documentation

6.5.2.1 subroutine CMISS::CMISS_FINALISE (INTEGER(INTG),intent(inout) *ERR*, TYPE(VARYING_STRING),intent(inout) *ERROR*, *)

Finalises CMISS.

Parameters:

ERR The error string

ERROR The error code

Definition at line 92 of file cmiss.f90.

References [BASE_ROUTINES::BASE_ROUTINES_FINALISE\(\)](#), [COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE\(\)](#), [COORDINATE_ROUTINES::COORDINATE_SYSTEMS_FINALISE\(\)](#), [PROBLEM_ROUTINES::PROBLEMS_FINALISE\(\)](#), and [REGION_ROUTINES::REGIONS_FINALISE\(\)](#).

Here is the call graph for this function:

6.5.2.2 subroutine CMISS::CMISS_INITIALISE (INTEGER(INTG),intent(inout) *ERR*, TYPE(VARYING_STRING),intent(inout) *ERROR*, *)

Initialises [CMISS](#).

Parameters:

ERR The error code

ERROR The error string

Definition at line 121 of file cmiss.f90.

References BASE_ROUTINES::BASE_ROUTINES_INITIALISE(), COMP_-
ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COORDINATE_-
ROUTINES::COORDINATE_SYSTEMS_INITIALISE(), PROBLEM_ROUTINES::PROBLEMS_-
FINALISE(), and REGION_ROUTINES::REGIONS_INITIALISE().

Here is the call graph for this function:

6.5.2.3 subroutine CMISS::CMISS_WRITE_ERROR (INTEGER(INTG),intent(inout) *ERR*, TYPE(VARYING_STRING),intent(inout) *ERROR*)

Writes the error string to screen.

Parameters:

ERR The error code

ERROR The error string

Definition at line 150 of file cmiss.f90.

References MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT.

6.5.3 Variable Documentation

6.5.3.1 INTEGER(INTG),parameter CMISS::CMISS_BUILD_VERSION = 1

Definition at line 73 of file cmiss.f90.

6.5.3.2 INTEGER(INTG),parameter CMISS::CMISS_MAJOR_VERSION = 0

Definition at line 71 of file cmiss.f90.

6.5.3.3 INTEGER(INTG),parameter CMISS::CMISS_MINOR_VERSION = 1

Definition at line 72 of file cmiss.f90.

6.6 CMISS_MPI Namespace Reference

This module contains [CMISS](#) MPI routines.

Functions

- subroutine [MPI_ERROR_CHECK](#) (ROUTINE, MPI_ERR_CODE, ERR, ERROR,*)

Checks to see if an MPI error has occurred during an MPI call and flags a [CMISS](#) error if it has.

6.6.1 Detailed Description

This module contains [CMISS](#) MPI routines.

6.6.2 Function Documentation

6.6.2.1 subroutine CMISS_MPI::MPI_ERROR_CHECK (CHARACTER(LEN=*) ROUTINE, INTEGER(INTG),intent(in) MPI_ERR_CODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Checks to see if an MPI error has occurred during an MPI call and flags a [CMISS](#) error if it has.

Parameters:

ROUTINE The name of the MPI routine that has just been called.

MPI_ERR_CODE The MPI error code returned from the MPI routine.

ERR The error code.

ERROR The error string

Definition at line 74 of file cmiss_mpi.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE(), COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(), FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS(), and FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.7 CMISS_PARMETIS Namespace Reference

This module is a [CMISS](#) buffer module to the ParMETIS library.

Classes

- interface [interface](#)

Functions

- subroutine [PARMETIS_PARTKWAY](#) (VERTEX_DISTANCE, XADJ, ADJNCY, VERTEX_WEIGHT, ADJ_WEIGHT, WEIGHT_FLA, NUM_FLAG, NCON,&NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartKway routine.

- subroutine [PARMETIS_PARTMESHKWAY](#) (ELEMENT_DISTANCE, ELEMENT_PTR, ELEMENT_INDEX, ELEMENT_WEIGHT, WEIGHT_FLAG, NUM_FLAG, CON,&NUMBER_COMMON_NODES, NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartMeshKway routine.

6.7.1 Detailed Description

This module is a [CMISS](#) buffer module to the ParMETIS library.

6.7.2 Function Documentation

- 6.7.2.1 subroutine CMISS_PARMETIS::PARMETIS_PARTKWAY**
`(INTEGER(INTG),dimension(:),intent(in) VERTEX_DISTANCE, INTEGER(INTG),dimension(:),intent(in) XADJ, INTEGER(INTG),dimension(:),intent(in) ADJNCY, INTEGER(INTG),dimension(:),intent(in) VERTEX_WEIGHT, INTEGER(INTG),dimension(:),intent(in) ADJ_WEIGHT, WEIGHT_FLA, INTEGER(INTG),intent(in) NUM_FLAG, INTEGER(INTG),intent(in) NCON, &,intent(in) NUMBER_PARTS, REAL(SP),dimension(:),intent(in) TP_WEIGHTS, REAL(SP),dimension(:),intent(in) UB_VEC, INTEGER(INTG),dimension(:),intent(in) OPTIONS, INTEGER(INTG),intent(out) NUMBER_EDGES_CUT, INTEGER(INTG),dimension(:),intent(out) PARTITION, INTEGER(INTG),intent(in) COMMUNICATOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the ParMetis ParMETIS_V3_PartKway routine.

Definition at line 120 of file cmiss_parmetis.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.7.2.2 subroutine CMISS_PARMETIS::PARMETIS_PARTMESHKWAY
(INTEGER(INTG),dimension(:),intent(in) *ELEMENT_DISTANCE*,
INTEGER(INTG),dimension(:),intent(in) *ELEMENT_PTR*,
INTEGER(INTG),dimension(:),intent(in) *ELEMENT_INDEX*,
INTEGER(INTG),dimension(:),intent(in) *ELEMENT_WEIGHT*,
INTEGER(INTG),intent(in) *WEIGHT_FLAG*, INTEGER(INTG),intent(in)
NUM_FLAG, CON, &,intent(in) *NUMBER_COMMON_NODES*,
INTEGER(INTG),intent(in) *NUMBER_PARTS*, REAL(SP),dimension(:),intent(in)
TP_WEIGHTS, REAL(SP),dimension(:),intent(in) *UB_VEC*,
INTEGER(INTG),dimension(:),intent(in) *OPTIONS*, INTEGER(INTG),intent(out)
NUMBER_EDGES_CUT, INTEGER(INTG),dimension(:),intent(out) *PARTITION*,
INTEGER(INTG),intent(in) *COMMUNICATOR*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the ParMetis ParMETIS_V3_PartMeshKway routine.

Definition at line 163 of file cmiss_parmetis.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Referenced by **MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8 CMISS_PETSC Namespace Reference

This module is a [CMISS](#) buffer module to the PETSc library.

Classes

- interface [interface](#)

Functions

- subroutine [PETSC_ERRORHANDLING_SET_OFF](#) (ERR, ERROR,*)

Set PETSc error handling on.
- subroutine [PETSC_ERRORHANDLING_SET_ON](#) (ERR, ERROR,*)

Set PETSc error handling on.
- subroutine [PETSC_FINALIZE](#) (ERR, ERROR,*)

Buffer routine to the PETSc PetscFinalize routine.
- subroutine [PETSC_INITIALIZE](#) (FILE, ERR, ERROR,*)

Buffer routine to the PETSc PetscInitialize routine.
- subroutine [PETSC_ISFINALISE](#) (IS_, ERR, ERROR,*)
 • subroutine [PETSC_ISINITIALISE](#) (IS_, ERR, ERROR,*)
 • subroutine [PETSC_ISDESTROY](#) (IS_, ERR, ERROR,*)

Buffer routine to the PETSc ISDestroy routine.
- subroutine [PETSC_ISLOCALTOGLOBALMAPPINGFINALISE](#) (ISLOCALTOGLOBALMAPPING_, ERR, ERROR,*)
 • subroutine [PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE](#) (ISLOCALTOGLOBALMAPPING_, ERR, ERROR,*)
 • subroutine [PETSC_ISLOCALTOGLOBALMAPPINGAPPLY](#) (CTX, TYPE, NIN, IDXIN, NOUT, IDXOUT, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.
- subroutine [PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS](#) (CTX, ISIN, ISOOUT, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.
- subroutine [PETSC_ISLOCALTOGLOBALMAPPINGCREATE](#) (COMMUNICATOR, N, GLOBALNUM, CTX, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.
- subroutine [PETSC_ISLOCALTOGLOBALMAPPINGDESTROY](#) (CTX, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.
- subroutine [PETSC_KSPCREATE](#) (COMMUNICATOR, KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPCreate routine.

- subroutine [PETSC_KSPDESTROY](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPDestroy routine.
- subroutine [PETSC_KSPGETCONVERGEDREASON](#) (KSP_, REASON, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetConvergedReason routine.
- subroutine [PETSC_KSPGETITERATIONNUMBER](#) (KSP_, ITERATION_NUMBER, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetIterationNumber routine.
- subroutine [PETSC_KSPGETPC](#) (KSP_, PC_, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetPC routine.
- subroutine [PETSC_KSPGETRESIDUALNORM](#) (KSP_, RESIDUAL_NORM, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetResidualNorm routine.
- subroutine [PETSC_KSPFINALISE](#) (KSP_, ERR, ERROR,*)
 • subroutine [PETSC_KSPINITIALISE](#) (KSP_, ERR, ERROR,*)
 • subroutine [PETSC_KSPSETFROMOPTIONS](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetFromOptions routine.
- subroutine [PETSC_KSPSETOPERATORS](#) (KSP_, AMAT, PMAT, FLAG, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetOperators routine.
- subroutine [PETSC_KSPSETTOLERANCES](#) (KSP_, RTOL, ATOL, DTOL, MAXITS, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetTolerances routine.
- subroutine [PETSC_KSPSETTYPE](#) (KSP_, METHOD, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetType routine.
- subroutine [PETSC_KSPSETUP](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetUp routine.
- subroutine [PETSC_KSPSOLVE](#) (KSP_, B, X, ERR, ERROR,*)

Buffer routine to the PETSc KSPSolve routine.
- subroutine [PETSC_LOGPRINTSUMMARY](#) (COMMUNICATOR, FILE, ERR, ERROR,*)

Buffer routine to the PETSc PetscLogPrintSummary routine.
- subroutine [PETSC_MATASSEMBLYBEGIN](#) (A, ASSEMBLY_TYPE, ERR, ERROR,*)

Buffer routine to the PETSc MatAssemblyBegin routine.
- subroutine [PETSC_MATASSEMBLYEND](#) (A, ASSEMBLY_TYPE, ERR, ERROR,*)

Buffer routine to the PETSc MatAssemblyEnd routine.
- subroutine [PETSC_MATCREATE](#) (COMMUNICATOR, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreate routine.

- subroutine **PETSC_MATCREATEMPIAIJ** (COMMUNICATOR, LOCAL_M, LOCAL_N, GLOB_L_M, GLOBAL_N, DIAG_NUMBER_NZ_PERROW, DIAG_NUMBER_NZ_EACHROW,&OFFDIAG_NUMBER_NZ_PERROW, OFFDIAG_NUMBER_NZ_EACHROW, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateMPIAIJ routine.

- subroutine **PETSC_MATCREATEMPIDENSE** (COMMUNICATOR, LOCAL_M, LOCAL_N, GLOBAL_M, GLOBAL_N, MATRIX_DATA, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateMPIDense routine.

- subroutine **PETSC_MATCREATESEQAIJ** (COMMUNICATOR, M, N, NUMBER_NZ_PERROW, NUMBER_NZ_EACHROW, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateSeqAIJ routine.

- subroutine **PETSC_MATCREATESEQDENSE** (COMMUNICATOR, M, N, MATRIX_DATA, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateSeqDense routine.

- subroutine **PETSC_MATDESTROY** (A, ERR, ERROR,*)

Buffer routine to the PETSc MatDestroy routine.

- subroutine **PETSC_MATGETARRAY** (A, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc MatGetArray routine.

- subroutine **PETSC_MATGETOWNERSHIPRANGE** (A, FIRST_ROW, LAST_ROW, ERR, ERROR,*)

Buffer routine to the PETSc MatGetOwnershipRange routine.

- subroutine **PETSC_MATGETVALUES** (A, M, M_INDICES, N, N_INDICES, VALUES, ERR, ERROR,*)

Buffer routine to the PETSc MatGetValues routine.

- subroutine **PETSC_MATFINALISE** (MAT_, ERR, ERROR,*)

- subroutine **PETSC_MATINITIALISE** (MAT_, ERR, ERROR,*)

- subroutine **PETSC_MATRESTOREARRAY** (A, ERR, ERROR,*)

Buffer routine to the PETSc MatRestoreArray routine.

- subroutine **PETSC_MATSETLOCALTOGLOBALMAPPING** (A, CTX, ERR, ERROR,*)

Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.

- subroutine **PETSC_MATSETOPTION** (A, OPTION, ERR, ERROR,*)

Buffer routine to the PETSc MatSetOption routine.

- subroutine **PETSC_MATSETSIZES** (A, LOCAL_M, LOCAL_N, GLOBAL_M, GLOBAL_N, ERR, ERROR,*)

Buffer routine to the PETSc MatSetSizes routine.

- subroutine **PETSC_MATSETVALUE** (A, ROW, COL, VALUE, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc MatSetValue routine.

- subroutine [PETSC_MATSETVALUES](#) (A, M, M_INDICES, N, N_INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc MatSetValues routine.
- subroutine [PETSC_MATSETVALUELOCAL](#) (A, ROW, COL, VALUE, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc MatSetValueLocal routine.
- subroutine [PETSC_MATSETVALUESLOCAL](#) (A, M, M_INDICES, N, N_INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc MatSetValuesLocal routine.
- subroutine [PETSC_MATVIEW](#) (A, V, ERR, ERROR,*)

Buffer routine to the PETSc MatView routine.
- subroutine [PETSC_MATZEROENTRIES](#) (A, ERR, ERROR,*)

Buffer routine to the PETSc MatZeroEntries routine.
- subroutine [PETSC_PCFINALISE](#) (PC_, ERR, ERROR,*)
 • subroutine [PETSC_PCINITIALISE](#) (PC_, ERR, ERROR,*)
 • subroutine [PETSC_PCSETTYPE](#) (PC_, METHOD, ERR, ERROR,*)

Buffer routine to the PETSc PCSetType routine.
- subroutine [PETSC_VECFINALISE](#) (VEC_, ERR, ERROR,*)
 • subroutine [PETSC_VEGINITIALISE](#) (VEC_, ERR, ERROR,*)
 • subroutine [PETSC_VECASSEMBLYBEGIN](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecAssemblyBegin routine.
- subroutine [PETSC_VECASSEMBLYEND](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecAssemblyEnd routine.
- subroutine [PETSC_VECCREATE](#) (COMMUNICATOR, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreate routine.
- subroutine [PETSC_VECCREATEHOST](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, NUMBER_GHOST, GHOSTS, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateGhost routine.
- subroutine [PETSC_VECCREATEHOSTWITHARRAY](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, NUMBER_GHOST, GHOSTS, ARRAY, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateGhostWithArray routine.
- subroutine [PETSC_VECCREATEMPI](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateMPI routine.
- subroutine [PETSC_VECCREATEMPIWITHARRAY](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, ARRAY, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateMPIWithArray routine.

- subroutine [PETSC_VECCREATESEQ](#) (COMMUNICATOR, SIZE, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateSeq routine.
- subroutine [PETSC_VECCREATESEQWITHARRAY](#) (COMMUNICATOR, SIZE, ARRAY, X, ERR, ERROR,*)

Buffer routine to the PETSc VecCreateSeqWithArray routine.
- subroutine [PETSC_VECDESTROY](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecDestroy routine.
- subroutine [PETSC_VECDUPLICATE](#) (OLD, NEW, ERR, ERROR,*)

Buffer routine to the PETSc VecDuplicate routine.
- subroutine [PETSC_VECGETARRAY](#) (X, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc VecGetArray routine.
- subroutine [PETSC_VECGETARRAYF90](#) (X, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc VecGetArrayF90 routine.
- subroutine [PETSC_VECGETLOCALSIZE](#) (X, SIZE, ERR, ERROR,*)

Buffer routine to the PETSc VecGetLocalSize routine.
- subroutine [PETSC_VECGETOWNERSHIPRANGE](#) (X, LOW, HIGH, ERR, ERROR,*)

Buffer routine to the PETSc VecGetOwnershipRange routine.
- subroutine [PETSC_VECGETSIZE](#) (X, SIZE, ERR, ERROR,*)

Buffer routine to the PETSc VecGetSize routine.
- subroutine [PETSC_VECGETVALUES](#) (X, N, INDICES, VALUES, ERR, ERROR,*)

Buffer routine to the PETSc VecGetValues routine.
- subroutine [PETSC_VECGHOSTGETLOCALFORM](#) (G, L, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostGetLocalForm routine.
- subroutine [PETSC_VECGHOSTRESTORELOCALFORM](#) (G, L, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostRestoreLocalForm routine.
- subroutine [PETSC_VECGHOSTUPDATEBEGIN](#) (X, INSERT_MODE, SCATTER_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostUpdateBegin routine.
- subroutine [PETSC_VECGHOSTUPDATEEND](#) (X, INSERT_MODE, SCATTER_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostUpdateEnd routine.
- subroutine [PETSC_VCRESTOREARRAY](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecRestoreArray routine.
- subroutine [PETSC_VCRESTOREARRAYF90](#) (X, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc VecRestoreArrayF90 routine.

- subroutine [PETSC_VECSET](#) (X, VALUE, ERR, ERROR,*)

Buffer routine to the PETSc VecSet routine.
- subroutine [PETSC_VECSETFROMOPTIONS](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecSetFromOptions routine.
- subroutine [PETSC_VECSETLOCALTOGLOBALMAPPING](#) (X, CTX, ERR, ERROR,*)

Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.
- subroutine [PETSC_VECSETVALUES](#) (X, N, INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetValues routine.
- subroutine [PETSC_VECSETVALUESLOCAL](#) (X, N, INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetValuesLocal routine.
- subroutine [PETSC_VECSETSIZES](#) (X, LOCAL_SIZE, GLOBAL_SIZE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetSizes routine.
- subroutine [PETSC_VECVIEW](#) (X, V, ERR, ERROR,*)

Buffer routine to the PETSc VecView routine.

Variables

- LOGICAL, save [PETSC_HANDLE_ERROR](#)

6.8.1 Detailed Description

This module is a [CMISS](#) buffer module to the PETSc library.

6.8.2 Function Documentation

6.8.2.1 subroutine CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Set PETSc error handling on.

Parameters:

ERR The error code

ERROR The error string

Definition at line 794 of file cmiss_petsc.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.8.2.2 subroutine CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Set PETSc error handling on.

Parameters:

ERR The error code

ERROR The error string

Definition at line 817 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.3 subroutine CMISS_PETSC::PETSC_FINALIZE (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc PetscFinalize routine.

Parameters:

ERR The error code

ERROR The error string

Definition at line 840 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.4 subroutine CMISS_PETSC::PETSC_INITIALIZE (CHARACTER(LEN=*),intent(in) *FILE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc PetscInitialize routine.

Parameters:

FILE Filename for PETSc options file

ERR The error code

ERROR The error string

Definition at line 869 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.5 subroutine CMISS_PETSC::PETSC_ISDESTROY (TYPE(PETSC_IS_TYPE),intent(in) *IS_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc ISDestroy routine.

Parameters:

IS_ The index set

ERR The error code

ERROR The error string

Definition at line 950 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.6 subroutine CMISS_PETSC::PETSC_ISFINALISE (TYPE(PETSC_IS_TYPE),intent(inout) *IS_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

IS_ The IS to finalise

ERR The error code

ERROR The error string

Definition at line 900 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.7 subroutine CMISS_PETSC::PETSC_ISINITIALISE (TYPE(PETSC_IS_TYPE),intent(inout) *IS_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

IS_ The IS to initialise

ERR The error code

ERROR The error string

Definition at line 926 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.8 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY
 (TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(in) *CTX*,
 INTEGER(INTG),intent(in) *TYPE*, INTEGER(INTG),intent(in) *NIN*,
 INTEGER(INTG),dimension(*),intent(in) *IDXIN*, INTEGER(INTG),intent(out) *NOUT*,
 INTEGER(INTG),dimension(*),intent(out) *IDXOUT*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.

Parameters:

CTX The local to global mapping context
TYPE The type of local to global mapping
NIN The number of local indicies
ERR The error code
ERROR The error string

Definition at line 1030 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.9 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS
 (TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(in) *CTX*,
 TYPE(PETSC_IS_TYPE),intent(in) *ISIN*, TYPE(PETSC_IS_TYPE),intent(out) *ISOUT*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.

Parameters:

CTX The local to global mapping context
ERR The error code
ERROR The error string

Definition at line 1065 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.10 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE
 (COMMUNICATOR, INTEGER(INTG),intent(in) *N*,
 INTEGER(INTG),dimension(*),intent(in) *GLOBALNUM*,
 TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(inout) *CTX*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.

Parameters:

N The number of local indices

GLOBALNUM The global number for each local index

CTX The local to global mapping context

ERR The error code

ERROR The error string

Definition at line 1097 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.11 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGDESTROY (TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(inout) *CTX*, *INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)*

Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.

Parameters:

CTX The local to global mapping context

ERR The error code

ERROR The error string

Definition at line 1130 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.12 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGFINALISE (TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(inout) *ISLOCALTOGLOBALMAPPING_, INTEGER(INTG),intent(out) ERR,* *TYPE(VARYING_STRING),intent(out) ERROR, *)*

Parameters:

ISLOCALTOGLOBALMAPPING_ The ISLocalToGlobalMapping to finalise

ERR The error code

ERROR The error string

Definition at line 980 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.13 subroutine CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE
(TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(inout)
ISLOCALTOGLOBALMAPPING_, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Parameters:

ISLOCALTOGLOBALMAPPING_ The ISLocalToGlobalMapping to initialise

ERR The error code

ERROR The error string

Definition at line 1006 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.14 subroutine CMISS_PETSC::PETSC_KSPCREATE (COMMUNICATOR,
TYPE(PETSC_KSP_TYPE),intent(inout) *KSP*_, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPCreate routine.

Parameters:

KSP_ On exit, the Ksp information

ERR The error code

ERROR The error string

Definition at line 1160 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.15 subroutine CMISS_PETSC::PETSC_KSPDESTROY (TYPE(PETSC_-KSP_TYPE),intent(inout) *KSP*_, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPDestroy routine.

Parameters:

KSP_ The Ksp to destroy

ERR The error code

ERROR The error string

Definition at line 1191 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.16 subroutine CMISS_PETSC::PETSC_KSPFINALISE (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

KSP_ The Ksp to finalise

ERR The error code

ERROR The error string

Definition at line 1345 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.17 subroutine CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, INTEGER(INTG),intent(out) *REASON*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc KSPGetConvergedReason routine.

Parameters:

KSP_ The KSP information

REASON On exit, the converged reason

ERR The error code

ERROR The error string

Definition at line 1221 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.18 subroutine CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, INTEGER(INTG),intent(out) *ITERATION_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Buffer routine to the PETSc KSPGetIterationNumber routine.

Parameters:

KSP_ The KSP information

ITERATION_NUMBER On exit, the number of iterations

ERR The error code

ERROR The error string

Definition at line 1252 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.19 subroutine CMISS_PETSC::PETSC_KSPGETPC (TYPE(PETSC_KSP_-
TYPE),intent(inout) *KSP_*, TYPE(PETSC_PC_TYPE),intent(inout) *PC_*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPGetPC routine.

Parameters:

KSP_ The Ksp to get the PC for

PC_ On exit, the PC associated with the Ksp

ERR The error code

ERROR The error string

Definition at line 1283 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.20 subroutine CMISS_PETSC::PETSC_KSPGETRESIDUALNORM (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, REAL(DP),intent(out) *RESIDUAL_NORM*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPGetResidualNorm routine.

Parameters:

KSP_ The Ksp to get the PC for

RESIDUAL_NORM On exit, the residual norm for the KSP

ERR The error code

ERROR The error string

Definition at line 1314 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.21 subroutine CMISS_PETSC::PETSC_KSPINITIALISE (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

KSP_ The Ksp to initialise

ERR The error code

ERROR The error string

Definition at line 1371 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.22 subroutine CMISS_PETSC::PETSC_KSPSETFROMOPTIONS (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc KSPSetFromOptions routine.

Parameters:

KSP_ The Ksp to set the options for

ERR The error code

ERROR The error string

Definition at line 1395 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.23 subroutine CMISS_PETSC::PETSC_KSPSETOPERATORS (TYPE(PETSC_KSP_-
TYPE),intent(inout) *KSP*_, TYPE(PETSC_MAT_TYPE),intent(inout) *AMAT*,
TYPE(PETSC_MAT_TYPE),intent(inout) *PMAT*, INTEGER(INTG),intent(in) *FLAG*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPSetOperators routine.

Parameters:

***KSP*_** The Ksp to set the operators for
AMAT The matrix associated with the linear system
PMAT The matrix to be used in constructing the preconditioner
FLAG Preconditioner matrix structure flag
ERR The error code
ERROR The error string

Definition at line 1425 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.24 subroutine CMISS_PETSC::PETSC_KSPSETTOLERANCES (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP*_, REAL(DP),intent(in) *RTOL*, REAL(DP),intent(in) *ATOL*, REAL(DP),intent(in) *DTOL*, INTEGER(INTG),intent(in) *MAXITS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc KSPSetTolerances routine.

Parameters:

***KSP*_** The Ksp to set the tolerances for
RTOL The relative tolerance to set
ATOL The absolute tolerance to set
DTOL The divergence tolerance to set
MAXITS The maximum number of iterations
ERR The error code
ERROR The error string

Definition at line 1458 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.25 subroutine CMISS_PETSC::PETSC_KSPSETTYPE (TYPE(PETSC_KSP_-
TYPE),intent(inout) *KSP*_, METHOD, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc KSPSetType routine.

Parameters:

***KSP*_** The Ksp to set the type for
ERR The error code
ERROR The error string

Definition at line 1492 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.26 subroutine CMISS_PETSC::PETSC_KSPSETUP (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP*_, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc KSPSetUp routine.

Parameters:

***KSP*_** The Ksp to set up
ERR The error code
ERROR The error string

Definition at line 1523 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.27 subroutine CMISS_PETSC::PETSC_KSPSOLVE (TYPE(PETSC_KSP_TYPE),intent(inout) *KSP*_, TYPE(PETSC_VEC_TYPE),intent(inout) *B*, TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc KSPSolve routine.

Parameters:

***KSP*_** The Ksp to set up
B The RHS vector
X The solution vector

ERR The error code

ERROR The error string

Definition at line 1553 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.28 subroutine CMISS_PETSC::PETSC_LOGPRINTSUMMARY (COMMUNICATOR, CHARACTER(LEN=*),intent(in) FILE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc PetscLogPrintSummary routine.

Parameters:

FILE Filename for the log summary

ERR The error code

ERROR The error string

Definition at line 1585 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.29 subroutine CMISS_PETSC::PETSC_MATASSEMBLYBEGIN (TYPE(PETSC_MAT_TYPE),intent(inout) A, ASSEMBLY_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatAssemblyBegin routine.

Parameters:

ERR The error code

ERROR The error string

Definition at line 1616 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.30 subroutine CMISS_PETSC::PETSC_MATASSEMBLYEND (TYPE(PETSC_MAT_TYPE),intent(inout) A, ASSEMBLY_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatAssemblyEnd routine.

Parameters:**A** The matrix to assemble**ERR** The error code**ERROR** The error string

Definition at line 1647 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.31 subroutine CMISS_PETSC::PETSC_MATCREATE (COMMUNICATOR, TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatCreate routine.

Parameters:**A** On exit, the created matrix**ERR** The error code**ERROR** The error string

Definition at line 1678 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.32 subroutine CMISS_PETSC::PETSC_MATCREATEMPIAIJ (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL_M, INTEGER(INTG),intent(in) LOCAL_N, GLOBAL_L_M, INTEGER(INTG),intent(in) GLOBAL_N, INTEGER(INTG),intent(in) DIAG_NUMBER_NZ_PERROW, INTEGER(INTG),dimension(*),intent(in) DIAG_NUMBER_NZ_EACHROW, &,intent(in) OFFDIAG_NUMBER_NZ_PERROW, INTEGER(INTG),dimension(*),intent(in) OFFDIAG_NUMBER_NZ_EACHROW, TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatCreateMPIAIJ routine.

Parameters:**LOCAL_M** The number of local rows**LOCAL_N** The number of local columns**GLOBAL_N** The number of global columns

DIAG_NUMBER_NZ_PERROW The maximum number of non-zeros per row in the diagonal part of the matrix

DIAG_NUMBER_NZ_EACHROW The number of non-zeros per row in the diagonal part of the matrix

OFFDIAG_NUMBER_NZ_EACHROW The number of non-zeros per row in the off-diagonal part of the matrix

A On exit, the matrix to create

ERR The error code

ERROR The error string

Definition at line 1709 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.33 subroutine CMISS_PETSC::PETSC_MATCREATEMPIENSE (COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL_M, INTEGER(INTG),intent(in) LOCAL_N, INTEGER(INTG),intent(in) GLOBAL_M, INTEGER(INTG),intent(in) GLOBAL_N, REAL(DP),dimension(*),intent(in) MATRIX_DATA, TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatCreateMPIDense routine.

Parameters:

LOCAL_M The number of local rows

LOCAL_N The number of local columns

GLOBAL_M The number of global columns

GLOBAL_N The number of global rows

MATRIX_DATA Optional, the allocated matrix data.

A On exit, the matrix to create

ERR The error code

ERROR The error string

Definition at line 1750 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.34 subroutine CMISS_PETSC::PETSC_MATCREATESEQAIJ (COMMUNICATOR, INTEGER(INTG),intent(in) M, INTEGER(INTG),intent(in) N, INTEGER(INTG),intent(in) NUMBER_NZ_PERROW, INTEGER(INTG),dimension(*),intent(in) NUMBER_NZ_EACHROW, TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc MatCreateSeqAIJ routine.

Parameters:

M The number of rows

N The number of columns

NUMBER_NZ_PERROW The maximum number of non-zeros per row

NUMBER_NZ_EACHROW The number of non-zeros in each row

A On exit, the created matrix

ERR The error code

ERROR The error string

Definition at line 1786 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.35 subroutine CMISS_PETSC::PETSC_MATCREATESEQDENSE
`(COMMUNICATOR, INTEGER(INTG),intent(in) M, INTEGER(INTG),intent(in) N, REAL(DP),dimension(*),intent(in) MATRIX_DATA, TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc MatCreateSeqDense routine.

Parameters:

M The number of rows

N The number of columns

MATRIX_DATA Optional, the allocated matrix data

A On exit, the created matrix

ERR The error code

ERROR The error string

Definition at line 1821 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.36 subroutine CMISS_PETSC::PETSC_MATDESTROY (`TYPE(PETSC_MAT_TYPE),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Buffer routine to the PETSc MatDestroy routine.

Parameters:

A The matrix to destroy

ERR The error code

ERROR The error string

Definition at line 1855 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.37 subroutine CMISS_PETSC::PETSC_MATFINALISE (TYPE(PETSC_-
MAT_TYPE),intent(inout) *MAT_*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

MAT_ The MAT to finalise
ERR The error code
ERROR The error string

Definition at line 1988 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.38 subroutine CMISS_PETSC::PETSC_MATGETARRAY (TYPE(PETSC_-
MAT_TYPE),intent(inout),target *A*, REAL(DP),dimension(:),pointer *ARRAY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatGetArray routine.

Parameters:

A The matrix to get the array for
ARRAY On exit, a pointer to the matrix array
ERR The error code
ERROR The error string

Definition at line 1885 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.39 subroutine CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE
(TYPE(PETSC_MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out)
FIRST_ROW, INTEGER(INTG),intent(out) *LAST_ROW*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatGetOwnershipRange routine.

Parameters:

A The matrix to get the ownership range of

FIRST_ROW On exit, the first row for the matrix

LAST_ROW On exit, the last row for the matrix

ERR The error code

ERROR The error string

Definition at line 1921 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.40 subroutine CMISS_PETSC::PETSC_MATGETVALUES (TYPE(PETSC_-
MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *M*,
INTEGER(INTG),dimension(*),intent(in) *M_INDICES*, INTEGER(INTG),intent(in)
N, INTEGER(INTG),dimension(*),intent(in) *N_INDICES*,
REAL(DP),dimension(*),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatGetValues routine.

Parameters:

A The matrix to get the values of

M The number of row indices

M_INDICES The row indices

N The number of column indices

N_INDICES The column indices

VALUES The values to get

ERR The error code

ERROR The error string

Definition at line 1953 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.41 subroutine CMISS_PETSC::PETSC_MATINITIALISE (TYPE(PETSC_-
MAT_TYPE),intent(inout) *MAT_*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Parameters:

MAT_ The MAT to initialise

ERR The error code

ERROR The error string

Definition at line 2014 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.42 subroutine CMISS_PETSC::PETSC_MATRESTOREARRAY
 (TYPE(PETSC_MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatRestoreArray routine.

Parameters:

- A*** The matrix to restore the array for
- ERR*** The error code
- ERROR*** The error string

Definition at line 2040 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.43 subroutine CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING
 (TYPE(PETSC_MAT_TYPE),intent(inout) *A*, TYPE(PETSC_-
 ISLOCALTOGLOBALMAPPING_TYPE),intent(in) *CTX*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.

Parameters:

- A*** The matrix to set the local to global mapping for
- CTX*** The local to global mapping context
- ERR*** The error code
- ERROR*** The error string

Definition at line 2070 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.44 subroutine CMISS_PETSC::PETSC_MATSETOPTION (TYPE(PETSC_-
 MAT_TYPE),intent(inout) *A*, *OPTION*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatSetOption routine.

Parameters:

- A*** The matrix to set the option for
- ERR*** The error code
- ERROR*** The error string

Definition at line 2101 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.45 subroutine CMISS_PETSC::PETSC_MATSETSIZES (TYPE(PETSC_MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *LOCAL_M*, INTEGER(INTG),intent(in) *LOCAL_N*, INTEGER(INTG),intent(in) *GLOBAL_M*, INTEGER(INTG),intent(in) *GLOBAL_N*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc MatSetSizes routine.

Parameters:

- A*** The matrix to set the size of
- LOCAL_M*** Number of local rows
- LOCAL_N*** Number of local columns
- GLOBAL_M*** Number of global rows
- GLOBAL_N*** Number of global columns
- ERR*** The error code
- ERROR*** The error string

Definition at line 2132 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.46 subroutine CMISS_PETSC::PETSC_MATSETVALUE (TYPE(PETSC_MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *ROW*, INTEGER(INTG),intent(in) *COL*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc MatSetValue routine.

Parameters:

- A*** The matrix to set the values of
- ROW*** The row index
- COL*** The column index
- VALUE*** The value to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2166 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.47 subroutine CMISS_PETSC::PETSC_MATSETVALUELOCAL (TYPE(PETSC_MAT_-
TYPE),intent(inout) *A*, INTEGER(INTG),intent(in) *ROW*, INTEGER(INTG),intent(in)
COL, REAL(DP),intent(in) *VALUE*, INSERT_MODE, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatSetValueLocal routine.

Parameters:

- A*** The matrix to set the values of
- ROW*** The row index
- COL*** The column index
- VALUE*** The value to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2236 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.48 subroutine CMISS_PETSC::PETSC_MATSETVALUES (TYPE(PETSC_-
MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in)
M, INTEGER(INTG),dimension(*),intent(in) *M_INDICES*,
INTEGER(INTG),intent(in) *N*, INTEGER(INTG),dimension(*),intent(in)
N_INDICES, REAL(DP),dimension(*),intent(in) *VALUES*, INSERT_MODE,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc MatSetValues routine.

Parameters:

- A*** The matrix to set the values of
- M*** The number of row indices
- M_INDICES*** The row indices
- N*** The number of column indices
- N_INDICES*** The column indices
- VALUES*** The values to set
- ERR*** The error code
- ERROR*** The error string

Definition at line 2200 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.49 subroutine CMISS_PETSC::PETSC_MATSETVALUESLOCAL
 (TYPE(PETSC_MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(in)
M, INTEGER(INTG),dimension(*),intent(in) *M_INDICES*,
 INTEGER(INTG),intent(in) *N*, INTEGER(INTG),dimension(*),intent(in)
N_INDICES, REAL(DP),dimension(*),intent(in) *VALUES*, INSERT_MODE,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc MatSetValuesLocal routine.

Parameters:

A The matrix to set the values of
M The number of row indices
M_INDICES The row indices
N The number of column indices
N_INDICES The column indices
VALUES The values to set
ERR The error code
ERROR The error string

Definition at line 2270 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.50 subroutine CMISS_PETSC::PETSC_MATVIEW (TYPE(PETSC_-MAT_TYPE),intent(inout) *A*, *V*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc MatView routine.

Parameters:

A The matrix to view
ERR The error code
ERROR The error string

Definition at line 2306 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.51 subroutine CMISS_PETSC::PETSC_MATZEROENTRIES (TYPE(PETSC_-MAT_TYPE),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc MatZeroEntries routine.

Parameters:

- A** The matrix to zero the entries of
- ERR** The error code
- ERROR** The error string

Definition at line 2337 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.52 subroutine CMISS_PETSC::PETSC_PCFINALISE (TYPE(PETSC_- PC_TYPE),intent(inout) PC_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

- PC_** The PC to finalise
- ERR** The error code
- ERROR** The error string

Definition at line 2367 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.8.2.53 subroutine CMISS_PETSC::PETSC_PCINITIALISE (TYPE(PETSC_- PC_TYPE),intent(inout) PC_, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

- PC_** The PC to initialise
- ERR** The error code
- ERROR** The error string

Definition at line 2393 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.54 subroutine CMISS_PETSC::PETSC_PCSETTYPE (TYPE(PETSC_PC_-
TYPE),intent(inout) *PC_*, METHOD, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc PCSetType routine.

Parameters:

PC_ The preconditioner to set the type of
ERR The error code
ERROR The error string

Definition at line 2417 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.8.2.55 subroutine CMISS_PETSC::PETSC_VECASSEMBLYBEGIN
(TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Buffer routine to the PETSc VecAssemblyBegin routine.

Parameters:

X The vector to begin the assembly of
ERR The error code
ERROR The error string

Definition at line 2500 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.56 subroutine CMISS_PETSC::PETSC_VECASSEMBLYEND (TYPE(PETSC_-
VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecAssemblyEnd routine.

Parameters:

X The vector to end the assembly of
ERR The error code
ERROR The error string

Definition at line 2530 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.57 subroutine CMISS_PETSC::PETSC_VECCREATE (COMMUNICATOR,
`TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`**

Buffer routine to the PETSc VecCreate routine.

Parameters:

`X` On exit, the created vector

`ERR` The error code

`ERROR` The error string

Definition at line 2560 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.58 subroutine CMISS_PETSC::PETSC_VECCREATEGHOST (COMMUNICATOR,
`INTEGER(INTG),intent(in) LOCAL_SIZE, INTEGER(INTG),intent(in)
GLOBAL_SIZE, INTEGER(INTG),intent(in) NUMBER_GHOST,`
`INTEGER(INTG),dimension(*),intent(in) GHOSTS, TYPE(PETSC_-
VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`**

Buffer routine to the PETSc VecCreateGhost routine.

Parameters:

`LOCAL_SIZE` The number of local elements

`GLOBAL_SIZE` The number of global elements

`NUMBER_GHOST` The number of ghost elements

`GHOSTS` The global location of the each ghost element

`X` On exit, the created vector

`ERR` The error code

`ERROR` The error string

Definition at line 2591 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.59 subroutine CMISS_PETSC::PETSC_VECCREATEGHOSTWITHARRAY
(COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL_SIZE,
INTEGER(INTG),intent(in) GLOBAL_SIZE, INTEGER(INTG),intent(in)
NUMBER_GHOST, INTEGER(INTG),dimension(*),intent(in) GHOSTS,
REAL(DP),dimension(*),intent(out) ARRAY, TYPE(PETSC_VEC_TYPE),intent(inout)
X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 \ast)

Buffer routine to the PETSc VecCreateGhostWithArray routine.

Parameters:

LOCAL_SIZE The number of local elements
GLOBAL_SIZE The number of global elements
NUMBER_GHOST The number of ghost elements
GHOSTS The global location of the each ghost element
ARRAY The preallocated array of matrix data
X On exit, the created vector
ERR The error code
ERROR The error string

Definition at line 2626 of file cmiss_petsc.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.8.2.60 subroutine CMISS_PETSC::PETSC_VECCREATEMPI(**COMMUNICATOR,**
INTEGER(INTG),intent(in) LOCAL_SIZE, INTEGER(INTG),intent(in) GLOBAL_SIZE,
TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, \ast)

Buffer routine to the PETSc VecCreateMPI routine.

Parameters:

LOCAL_SIZE The number of local elements
GLOBAL_SIZE The number of global elements
X On exit, the created vector
ERR The error code
ERROR The error string

Definition at line 2662 of file cmiss_petsc.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.8.2.61 subroutine CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY
(COMMUNICATOR, INTEGER(INTG),intent(in) LOCAL_SIZE,
INTEGER(INTG),intent(in) GLOBAL_SIZE, REAL(DP),dimension(*),intent(out)
ARRAY, TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc VecCreateMPIWithArray routine.

Parameters:

LOCAL_SIZE The number of local elements
GLOBAL_SIZE The number of global elements
ARRAY The preallocated array for the vector data
X On exit, the created vector
ERR The error code
ERROR The error string

Definition at line 2695 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.62 subroutine CMISS_PETSC::PETSC_VECCREATESEQ (COMMUNICATOR,
INTEGER(INTG),intent(in) SIZE, TYPE(PETSC_VEC_TYPE),intent(inout) X,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc VecCreateSeq routine.

Parameters:

SIZE The size of the vector
X On exit, the created vector
ERR The error code
ERROR The error string

Definition at line 2729 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.63 subroutine CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY
(COMMUNICATOR, INTEGER(INTG),intent(in) SIZE,
REAL(DP),dimension(*),intent(out) ARRAY, TYPE(PETSC_VEC_TYPE),intent(inout)
X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***)**

Buffer routine to the PETSc VecCreateSeqWithArray routine.

Parameters:

SIZE The size of the vector
ARRAY The preallocated array for the vector data
X On exit, the created vector
ERR The error code
ERROR The error string

Definition at line 2761 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.64 subroutine CMISS_PETSC::PETSC_VECDESTROY (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecDestroy routine.

Parameters:

X The vector to destroy
ERR The error code
ERROR The error string

Definition at line 2794 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.65 subroutine CMISS_PETSC::PETSC_VECDUPLICATE (TYPE(PETSC_VEC_TYPE),intent(inout) *OLD*, TYPE(PETSC_VEC_TYPE),intent(out) *NEW*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecDuplicate routine.

Parameters:

OLD The vector to duplicate
NEW On exit, the new duplicated vector
ERR The error code
ERROR The error string

Definition at line 2824 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.66 subroutine CMISS_PETSC::PETSC_VECFINALISE (TYPE(PETSC_-
VEC_TYPE),intent(inout) *VEC_*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Parameters:

VEC_ The Vec to finalise

ERR The error code

ERROR The error string

Definition at line 2448 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.67 subroutine CMISS_PETSC::PETSC_VECGETARRAY (TYPE(PETSC_-
VEC_TYPE),intent(inout),target *X*, REAL(DP),dimension(:),pointer *ARRAY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecGetArray routine.

Parameters:

X The vector to get the array of

ARRAY On exit, a pointer to the array of the vector

ERR The error code

ERROR The error string

Definition at line 2855 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.68 subroutine CMISS_PETSC::PETSC_VECGETARRAYF90 (TYPE(PETSC_-
VEC_TYPE),intent(inout),target *X*, REAL(DP),dimension(:),pointer *ARRAY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecGetArrayF90 routine.

Parameters:

X The vector to get the array of

ARRAY On exit, a pointer to the array of the vector

ERR The error code

ERROR The error string

Definition at line 2891 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.69 subroutine CMISS_PETSC::PETSC_VECGETLOCALSIZE (TYPE(PETSC_-
VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *SIZE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecGetLocalSize routine.

Parameters:

X The vector to get the local size of
SIZE On exit, the local size of the vector
ERR The error code
ERROR The error string

Definition at line 2926 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.70 subroutine CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE
(TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out)
LOW, INTEGER(INTG),intent(out) *HIGH*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecGetOwnershipRange routine.

Parameters:

X The vector to get the ownership range of
LOW On exit, the low end of the range
HIGH On exit, the high end of the range
ERR The error code
ERROR The error string

Definition at line 2957 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.8.2.71 subroutine CMISS_PETSC::PETSC_VECGETSIZE (TYPE(PETSC_-
VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *SIZE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecGetSize routine.

Parameters:

X The vector to get the size of
SIZE On exit, the size of the vector
ERR The error code

ERROR The error string

Definition at line 2989 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.72 subroutine CMISS_PETSC::PETSC_VECGETVALUES (`TYPE(PETSC_VEC_TYPE),intent(inout) X`, `INTEGER(INTG),intent(in) N`, `INTEGER(INTG),dimension(*),intent(in) INDICES`, `REAL(DP),dimension(*),intent(out) VALUES`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Buffer routine to the PETSc VecGetValues routine.

Parameters:

X The vector to set the values for

N The number of indices to get

INDICES The indices to get

VALUES On return, the values at the specified indices

ERR The error code

ERROR The error string

Definition at line 3020 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.73 subroutine CMISS_PETSC::PETSC_VECGHOSTGETLOCALFORM (`TYPE(PETSC_VEC_TYPE),intent(inout) G`, `TYPE(PETSC_VEC_TYPE),intent(inout) L`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Buffer routine to the PETSc VecGhostGetLocalForm routine.

Parameters:

G The global form of the vector

L On exit, the local form of the vector with ghosts

ERR The error code

ERROR The error string

Definition at line 3053 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.74 subroutine CMISS_PETSC::PETSC_VECHOSTRESTORELOCALFORM
 (TYPE(PETSC_VEC_TYPE),intent(inout) *G*, TYPE(PETSC_VEC_TYPE),intent(inout) *L*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecGhostRestoreLocalForm routine.

Parameters:

- G* The global form of the vector
- L* The local form of the vector
- ERR* The error code
- ERROR* The error string

Definition at line 3084 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.75 subroutine CMISS_PETSC::PETSC_VECHOSTUPDATEBEGIN
 (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INSERT_MODE, SCATTER_MODE,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecGhostUpdateBegin routine.

Parameters:

- X* The vector to begin the ghost update for
- ERR* The error code
- ERROR* The error string

Definition at line 3115 of file cmiss_petsc.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.8.2.76 subroutine CMISS_PETSC::PETSC_VECHOSTUPDATEEND
 (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INSERT_MODE, SCATTER_MODE,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecGhostUpdateEnd routine.

Parameters:

- X* The vector to end the ghost update for
- ERR* The error code
- ERROR* The error string

Definition at line 3147 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.77 subroutine CMISS_PETSC::PETSC_VECINITIALISE (TYPE(PETSC_VEC_TYPE),intent(inout) *VEC_*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

VEC_ The Vec to initialise
ERR The error code
ERROR The error string

Definition at line 2474 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.78 subroutine CMISS_PETSC::PETSC_VECRESTOREARRAY (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecRestoreArray routine.

Parameters:

X The vector to restore the array of
ERR The error code
ERROR The error string

Definition at line 3179 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.79 subroutine CMISS_PETSC::PETSC_VECRESTOREARRAYF90 (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, REAL(DP),dimension(:),pointer *ARRAY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecRestoreArrayF90 routine.

Parameters:

X The vector to restore the array of
ARRAY A pointer to the data to restore
ERR The error code

ERROR The error string

Definition at line 3209 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.80 subroutine CMISS_PETSC::PETSC_VECSET (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecSet routine.

Parameters:

X The vector to set the value of

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3240 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.81 subroutine CMISS_PETSC::PETSC_VECSETFROMOPTIONS (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecSetFromOptions routine.

Parameters:

X The vector to set the options for

ERR The error code

ERROR The error string

Definition at line 3271 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.82 subroutine CMISS_PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING (TYPE(PETSC_VEC_TYPE),intent(inout) *X*, TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE),intent(in) *CTX*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.

Parameters:

X The vector to set the local to global mapping for

CTX The local to global mapping context

ERR The error code

ERROR The error string

Definition at line 3301 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.83 subroutine CMISS_PETSC::PETSC_VECSETSIZES (TYPE(PETSC_-
VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(in) *LOCAL_SIZE*,
INTEGER(INTG),intent(in) *GLOBAL_SIZE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecSetSizes routine.

Parameters:

X The vector to set the sizes of

LOCAL_SIZE The number of local elements

GLOBAL_SIZE The number of global elements

ERR The error code

ERROR The error string

Definition at line 3400 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.8.2.84 subroutine CMISS_PETSC::PETSC_VECSETVALUES (TYPE(PETSC_-
VEC_TYPE),intent(inout) *X*, INTEGER(INTG),intent(in) *N*,
INTEGER(INTG),dimension(*),intent(in) *INDICES*, REAL(DP),dimension(*),intent(in)
VALUES, *INSERT_MODE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Buffer routine to the PETSc VecSetValues routine.

Parameters:

X The vector to set the values for

N The number of indicies

INDICES The indices

VALUES The values to set

ERR The error code

ERROR The error string

Definition at line 3332 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.85 subroutine CMISS_PETSC::PETSC_VECSETVALUESLOCAL
`(TYPE(PETSC_VEC_TYPE),intent(inout) X, INTEGER(INTG),intent(in) N,`
`INTEGER(INTG),dimension(*),intent(in) INDICES, REAL(DP),dimension(*),intent(in)`
`VALUES, INSERT_MODE, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Buffer routine to the PETSc VecSetValuesLocal routine.

Parameters:

`X` The vector to set the values of
`N` The number of indices
`INDICES` The local indices
`VALUES` The values to set
`ERR` The error code
`ERROR` The error string

Definition at line 3366 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.2.86 subroutine CMISS_PETSC::PETSC_VECVIEW (TYPE(PETSC_VEC_TYPE),intent(inout) X, V, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Buffer routine to the PETSc VecView routine.

Parameters:

`X` The vector to view
`ERR` The error code
`ERROR` The error string

Definition at line 3432 of file cmiss_petsc.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.8.3 Variable Documentation

6.8.3.1 LOGICAL,save CMISS_PETSC::PETSC_HANDLE_ERROR

Definition at line 164 of file cmiss_petsc.f90.

6.9 CMISS_PETSC_TYPES Namespace Reference

This module contains types related to the PETSc library.

Classes

- struct [PETSC_IS_TYPE](#)
- struct [PETSC_ISLOCALTOGLOBALMAPPING_TYPE](#)
- struct [PETSC_KSP_TYPE](#)
- struct [PETSC_MAT_TYPE](#)
- struct [PETSC_PC_TYPE](#)
- struct [PETSC_SNES_TYPE](#)
- struct [PETSC_VEC_TYPE](#)

6.9.1 Detailed Description

This module contains types related to the PETSc library.

6.10 COMP_ENVIRONMENT Namespace Reference

This module contains all computational environment variables.

Classes

- struct [CACHE_TYPE](#)
Contains information on a cache hierarchy.
- struct [COMPUTATIONAL_NODE_TYPE](#)
Contains information on a computational node containing a number of processors.
- struct [MPI_COMPUTATIONAL_NODE_TYPE](#)
Contains information on the MPI type to transfer information about a computational node.
- struct [COMPUTATIONAL_ENVIRONMENT_TYPE](#)
Contains information on the computational environment the program is running in.

Functions

- subroutine [COMPUTATIONAL_NODE_FINALISE](#) (COMPUTATIONAL_NODE, ERR, ERROR,*)
Finalises the computational node data structures and deallocates all memory.
- subroutine [COMPUTATIONAL_NODE_INITIALISE](#) (COMPUTATIONAL_NODE, RANK, ERR, ERROR,*)
Initialises the computational node data structures.
- subroutine [COMPUTATIONAL_NODE_MPI_TYPE_FINALISE](#) (ERR, ERROR,*)
Finalises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).
- subroutine [COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE](#) (COMPUTATIONAL_NODE, ERR, ERROR,*)
Initialises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).
- subroutine [COMPUTATIONAL_ENVIRONMENT_FINALISE](#) (ERR, ERROR,*)
Finalises the computational environment data structures and deallocates all memory.
- subroutine [COMPUTATIONAL_ENVIRONMENT_INITIALISE](#) (ERR, ERROR,*)
Initialises the computational environment data structures.
- INTEGER(INTG) [COMPUTATIONAL_NODE_NUMBER_GET](#) (ERR, ERROR)
Returns the number/rank of the computational nodes.
- INTEGER(INTG) [COMPUTATIONAL_NODES_NUMBER_GET](#) (ERR, ERROR)
Returns the number of computational nodes.

Variables

- TYPE(COMPUTATIONAL_ENVIRONMENT_TYPE) COMPUTATIONAL_ENVIRONMENT
The computational environment the program is running in.
- TYPE(MPI_COMPUTATIONAL_NODE_TYPE) MPI_COMPUTATIONAL_NODE_TYPE_-
DATA
The MPI data on the computational nodes.

6.10.1 Detailed Description

This module contains all computational environment variables.

6.10.2 Function Documentation

6.10.2.1 subroutine COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_- FINALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Finalises the computational environment data structures and deallocates all memory.

Parameters:

ERR The error code

ERROR The error string

Definition at line 278 of file computational_environment.f90.

References COMPUTATIONAL_ENVIRONMENT, COMPUTATIONAL_NODE_FINALISE(), COMPUTATIONAL_NODE_MPI_TYPE_FINALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), CMISS_MPI::MPI_ERROR_CHECK(), and CMISS_PETSC::PETSC_FINALIZE().

Referenced by CMISS::CMISS_FINALISE(), and COMPUTATIONAL_ENVIRONMENT_-INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.2 subroutine COMP_ENVIRONMENT::COMPUTATIONAL_- ENVIRONMENT_INITIALISE (INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Initialises the computational environment data structures.

Parameters:

ERR The error code

ERROR The error string

Definition at line 322 of file computational_environment.f90.

References COMPUTATIONAL_ENVIRONMENT, COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMPUTATIONAL_NODE_INITIALISE(), COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE(), BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::DIAGNOSTICS2, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MPI_COMPUTATIONAL_NODE_TYPE_DATA, CMISS_MPI::MPI_ERROR_CHECK(), and CMISS_PETSC::PETSC_INITIALIZE().

Referenced by CMISS::CMISS_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.3 subroutine COMPUTATIONALENVIRONMENT::COMPUTATIONAL_NODE_FINALISE
`(TYPE(COMPUTATIONAL_NODE_TYPE),intent(inout) COMPUTATIONAL_NODE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Finalises the computational node data structures and deallocates all memory.

Parameters:

COMPUTATIONAL_NODE The computational node to finalise
ERR The error code
ERROR The error string

Definition at line 116 of file computational_environment.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COMPUTATIONAL_ENVIRONMENT_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.4 subroutine COMPUTATIONALENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE
`(TYPE(COMPUTATIONAL_NODE_TYPE),intent(out) COMPUTATIONAL_NODE,
 INTEGER(INTG),intent(in) RANK, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the computational node data structures.

Parameters:

COMPUTATIONAL_NODE The computational node to initialise
RANK The MPI rank of the computational node
ERR The error code
ERROR The error string

Definition at line 143 of file computational_environment.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.5 subroutine `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE` (`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Finalises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).

Parameters:

ERR The error code

ERROR The error string

Definition at line 174 of file computational_environment.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MPI_COMPUTATIONAL_NODE_TYPE_DATA`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_FINALISE()`, and `COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.6 subroutine `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE` (`TYPE(COMPUTATIONAL_NODE_TYPE),intent(in) COMPUTATIONAL_NODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Initialises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).

Parameters:

COMPUTATIONAL_NODE The computational node containing the MPI type to initialise

ERR The error code

ERROR The error string

Definition at line 208 of file computational_environment.f90.

References `COMPUTATIONAL_NODE_MPI_TYPE_FINALISE()`, `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS3`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MPI_COMPUTATIONAL_NODE_TYPE_DATA`, and `CMISS_MPI::MPI_ERROR_CHECK()`.

Referenced by `COMPUTATIONAL_ENVIRONMENT_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.7 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_- GET (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*)

Returns the number/rank of the computational nodes.

Parameters:

ERR The error code

ERROR The error string

Definition at line 405 of file computational_environment.f90.

References COMPUTATIONAL_ENVIRONMENT, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), and FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT().

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.2.8 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_- NODES_NUMBER_GET (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*)

Returns the number of computational nodes.

Parameters:

ERR The error code

ERROR The error string

Definition at line 434 of file computational_environment.f90.

References COMPUTATIONAL_ENVIRONMENT, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_SET(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT(), FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT(), and FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT().

Here is the call graph for this function:

Here is the caller graph for this function:

6.10.3 Variable Documentation

6.10.3.1 TYPE(COMPUTATIONAL_ENVIRONMENT_TYPE) COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT

The computational environment the program is running in.

Definition at line 97 of file computational_environment.f90.

Referenced by COMPUTATIONAL_ENVIRONMENT_FINALISE(), COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMPUTATIONAL_NODE_NUMBER_GET(), COMPUTATIONAL_NODES_NUMBER_GET(), MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), and SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH().

6.10.3.2 TYPE(MPI_COMPUTATIONAL_NODE_TYPE) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE_DATA

The MPI data on the computational nodes.

Definition at line 98 of file computational_environment.f90.

Referenced by COMPUTATIONAL_ENVIRONMENT_INITIALISE(), COMPUTATIONAL_NODE_MPI_TYPE_FINALISE(), and COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE().

6.11 COORDINATE_ROUTINES Namespace Reference

This module contains all coordinate transformation and support routines.

Classes

- struct [COORDINATE_SYSTEM_PTR_TYPE](#)
- struct [COORDINATE_SYSTEMS_TYPE](#)
- interface [COORDINATE_CONVERT_FROM_RC](#)
- interface [COORDINATE_CONVERT_TO_RC](#)
- interface [COORDINATE_DELTA_CALCULATE](#)
- interface [COORDINATE_SYSTEM_DIMENSION_SET](#)
- interface [COORDINATE_SYSTEM_FOCUS_SET](#)
- interface [COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET](#)
- interface [COORDINATE_SYSTEM_TYPE_SET](#)
- interface [COORDINATE_SYSTEM_ORIGIN_SET](#)
- interface [COORDINATE_SYSTEM_ORIENTATION_SET](#)
- interface [DXZ](#)
- interface [D2ZX](#)
- interface [DZX](#)
- interface [COORDINATE_DERIVATIVE_CONVERT_TO_RC](#)
- interface [COORDINATE_SYSTEM_DESTROY](#)

Functions

- REAL(DP) [COORDINATE_CONVERT_FROM_RC_DP](#) (COORDINATE_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE_CONVERT_FROM_RC_SP](#) (COORDINATE_SYSTEM, Z, ERR, ER-ROR)
- REAL(DP) [COORDINATE_CONVERT_TO_RC_DP](#) (COORDINATE_SYSTEM, X, ERR, ER-ROR)
- REAL(SP) [COORDINATE_CONVERT_TO_RC_SP](#) (COORDINATE_SYSTEM, X, ERR, ER-ROR)
- REAL(DP) [COORDINATE_DELTA_CALCULATE_DP](#) (COORDINATE_SYSTEM, X, Y, ERR, ERROR)
- subroutine [COORDINATE_METRICS_CALCULATE](#) (COORDINATE_SYSTEM, JACOBIAN_-TYPE, METRICS, ERR, ERROR,*)
- subroutine [COORDINATE_SYSTEM_NORMAL_CALCULATE](#) (COORDINATE_SYSTEM, RE-VERSE, X, N, ERR, ERROR,*)
- INTEGER(INTG) [COORDINATE_SYSTEM_DIMENSION_GET](#) (COORDINATE_SYSTEM)
- REAL(DP) [COORDINATE_SYSTEM_FOCUS_GET](#) (COORDINATE_SYSTEM, ERR, ERROR)
- INTEGER(INTG) [COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_GET](#) (COORDINATE_SYSTEM)
- INTEGER(INTG) [COORDINATE_SYSTEM_TYPE_GET](#) (COORDINATE_SYSTEM)
- subroutine [COORDINATE_SYSTEM_DIMENSION_SET_NUMBER](#) (USER_NUMBER, DI-MENSION, ERR, ERROR,*)
- subroutine [COORDINATE_SYSTEM_DIMENSION_SET_PTR](#) (COORDINATE_SYSTEM, DI-MENSION, ERR, ERROR,*)
- subroutine [COORDINATE_SYSTEM_FOCUS_SET_NUMBER](#) (USER_NUMBER, FOCUS, ERR, ERROR,*)

- subroutine `COORDINATE_SYSTEM_FOCUS_SET_PTR` (`COORDINATE_SYSTEM`, `FOCUS`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER` (`USER_NUMBER`, `RADIAL_INTERPOLATION_TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR` (`COORDINATE_SYSTEM`, `RADIAL_INTERPOLATION_TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_TYPE_SET_NUMBER` (`USER_NUMBER`, `TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_TYPE_SET_PTR` (`COORDINATE_SYSTEM`, `TYPE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_NUMBER` (`USER_NUMBER`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIGIN_SET_PTR` (`COORDINATE_SYSTEM`, `ORIGIN`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER` (`USER_NUMBER`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_PTR` (`COORDINATE_SYSTEM`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_CREATE_START` (`USER_NUMBER`, `COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_CREATE_FINISH` (`COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DESTROY_NUMBER` (`USER_NUMBER`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DESTROY_PTR` (`COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- REAL(DP) `DXZ_DP` (`COORDINATE_SYSTEM`, `I`, `X`, `ERR`, `ERROR`)
- REAL(DP) `D2ZX_DP` (`COORDINATE_SYSTEM`, `I`, `J`, `X`, `ERR`, `ERROR`)
- REAL(DP) `DZX_DP` (`COORDINATE_SYSTEM`, `I`, `X`, `ERR`, `ERROR`)
- subroutine `CO` (`COORDINATE_SYSTEM`, `PART_DERIV_TYPE`, `X`, `Z,&ERR`, `ERROR,*`)
- subroutine `CO` (`COORDINATE_SYSTEM`, `PART_DERIV_TYPE`, `X`, `Z,&ERR`, `ERROR,*`)
- subroutine `COORDINATE_DERIVATIVE_NORM` (`COORDINATE_SYSTEM`, `PART_DERIV_INDEX`, `INTERPOLATED_POINT`, `DERIV_NORM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_INTERPOLATION_ADJUST` (`COORDINATE_SYSTEM`, `PARTIAL_DERIVATIVE_INDEX`, `VALUE`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_INTERPOLATION_PARAMETERS_ADJUST` (`COORDINATE_SYSTEM`, `INTERPOLATION_PARAMETERS`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_USER_NUMBER_FIND` (`USER_NUMBER`, `COORDINATE_SYSTEM`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEMS_FINALISE` (`ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEMS_INITIALISE` (`ERR`, `ERROR,*`)

Variables

- INTEGER(INTG), parameter `COORDINATE_RECTANGULAR_CARTESIAN_TYPE` = 1
Rectangular Cartesian coordinate system type.
- INTEGER(INTG), parameter `COORDINATE_CYCLINDRICAL_POLAR_TYPE` = 2
Cylindrical polar coordinate system type.
- INTEGER(INTG), parameter `COORDINATE_SPHERICAL_POLAR_TYPE` = 3

Spherical polar coordinate system type.

- INTEGER(INTG), parameter **COORDINATE_PROLATE_SPHEROIDAL_TYPE** = 4
Prolate spheroidal coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_OBLATE_SPHEROIDAL_TYPE** = 5
Oblate spheroidal coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_NO_RADIAL_INTERPOLATION_TYPE** = 0
No radial interpolation.
- INTEGER(INTG), parameter **COORDINATE_RADIAL_INTERPOLATION_TYPE** = 1
r radial interpolation
- INTEGER(INTG), parameter **COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE** = 2
 r^2 radial interpolation
- INTEGER(INTG), parameter **COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE** = 3
 r^3 radial interpolation
- INTEGER(INTG), parameter **COORDINATE_JACOBIAN_LINE_TYPE** = 1
Line type Jacobian.
- INTEGER(INTG), parameter **COORDINATE_JACOBIAN_AREA_TYPE** = 2
Area type Jacobian.
- INTEGER(INTG), parameter **COORDINATE_JACOBIAN_VOLUME_TYPE** = 3
Volume type Jacobian.
- CHARACTER(LEN=21) **COORDINAT**
- CHARACTER(LEN=21) **_SYSTEM_TYPE_STRING** = & (/ , & , & , & , & /)
- TYPE(**COORDINATE_SYSTEMS_TYPE**) **COORDINATE_SYSTEMS**
- TYPE(**COORDINATE_SYSTEM_TYPE**), pointer **GLOBAL_COORDINATE_SYSTEM**

6.11.1 Detailed Description

This module contains all coordinate transformation and support routines.

6.11.2 Function Documentation

6.11.2.1 subroutine **COORDINATE_ROUTINES::CO** (TYPE(**COORDINATE_SYSTEM_TYPE**),intent(in) **COORDINATE_SYSTEM**, INTEGER(INTG),intent(in) **PART_DERIV_TYPE**, REAL(SP),dimension(:, :, :),intent(in) **X**, REAL(SP),dimension(:, :),intent(out) **Z**, &,intent(out) **ERR**, TYPE(**VARYING_STRING**),intent(out) **ERROR**, *) [private]

Definition at line 3085 of file coordinate_routines.f90.

References KINDS::SP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.2 subroutine COORDINATE_ROUTINES::CO (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, INTEGER(INTG),intent(in) PART_DERIV_TYPE, REAL(DP),dimension(:, :, :),intent(in) X, REAL(DP),dimension(:, :),intent(out) Z, &,intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 2516 of file coordinate_routines.f90.

References KINDS::DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.3 REAL(DP) COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, REAL(DP),dimension(:, :),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 227 of file coordinate_routines.f90.

References KINDS::DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.4 REAL(SP) COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, REAL(SP),dimension(:, :),intent(in) Z, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 352 of file coordinate_routines.f90.

References KINDS::SP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.5 REAL(DP) COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP
 (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,
 REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Definition at line 488 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.6 REAL(SP) COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP
 (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,
 REAL(SP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Definition at line 571 of file coordinate_routines.f90.

References KINDS::_SP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.7 REAL(DP) COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP
 (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,
 REAL(DP),dimension(:),intent(in) X, REAL(DP),dimension(:),intent(in) Y,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
 [private]**

Definition at line 664 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.8 subroutine COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM
 (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
 INTEGER(INTG),intent(in) PART_DERIV_INDEX, TYPE(FIELD_-
 INTERPOLATED_POINT_TYPE),pointer INTERPOLATED_POINT,
 REAL(DP),intent(out) DERIV_NORM, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 3654 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_-

RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.9 subroutine COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM, INTEGER(INTG),intent(in) PARTIAL_DERIVATIVE_INDEX, REAL(DP),intent(inout) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 3752 of file coordinate_routines.f90.

References KINDS::DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE, COORDINATE_RADIAL_INTERPOLATION_TYPE, COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATE_GAUSS(), and FIELD_ROUTINES::FIELD_INTERPOLATE_XI().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.10 subroutine COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM, TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer INTERPOLATION_PARAMETERS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 3864 of file coordinate_routines.f90.

References COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE, COORDINATE_RADIAL_INTERPOLATION_TYPE, COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.11 subroutine COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE
`(TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
 INTEGER(INTG),intent(in) JACOBIAN_TYPE, TYPE(FIELD_INTERPOLATED_-
 POINT_METRICS_TYPE),pointer METRICS, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 721 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_-JACOBIAN_AREA_TYPE, COORDINATE_JACOBIAN_LINE_TYPE, COORDINATE_JACOBIAN_-VOLUME_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_-SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_-SPHERICAL_POLAR_TYPE, BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_-INDICES.

Referenced by FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.12 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_-
`FINISH (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Definition at line 1772 of file coordinate_routines.f90.

References COORDINATE_SYSTEMS, BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.11.2.13 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(COORDINATE_SYSTEM_-
 TYPE),pointer COORDINATE_SYSTEM, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1697 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_NO_RADIAL_INTERPOLATION_TYPE, COORDINATE_-RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.11.2.14 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_-
`NUMBER (INTEGER(INTG) USER_NUMBER, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 1818 of file coordinate_routines.f90.

References COORDINATE_SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.11.2.15 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_PTR
(TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Definition at line 1874 of file coordinate_routines.f90.

References COORDINATE_SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.11.2.16 INTEGER(INTG) COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
DIMENSION_GET (TYPE(COORDINATE_SYSTEM_TYPE),intent(in)
COORDINATE_SYSTEM)**

Definition at line 1074 of file coordinate_routines.f90.

**6.11.2.17 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
DIMENSION_SET_NUMBER (INTEGER(INTG),intent(in) USER_NUMBER,
INTEGER(INTG),intent(in) DIMENSION, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Definition at line 1180 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_DIMENSION_SET_PTR(), COORDINATE_SYSTEM_USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.18 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_-
SET_PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
INTEGER(INTG),intent(in) DIMENSION, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Definition at line 1211 of file coordinate_routines.f90.

References COORDINATE_CYLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_DIMENSION_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.19 REAL(DP) COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET
(TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)**

Definition at line 1097 of file coordinate_routines.f90.

References COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.20 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_-
NUMBER (INTEGER(INTG),intent(in) USER_NUMBER, REAL(DP),intent(in)
FOCUS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]**

Definition at line 1281 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_FOCUS_SET_PTR(), COORDINATE_SYSTEM_USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.21 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
FOCUS_SET_PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer
COORDINATE_SYSTEM, REAL(DP),intent(in) FOCUS, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Definition at line 1312 of file coordinate_routines.f90.

References COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_FOCUS_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.22 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
NORMAL_CALCULATE (TYPE(COORDINATE_SYSTEM_TYPE),pointer
COORDINATE_SYSTEM, LOGICAL,intent(in) REVERSE,
REAL(DP),dimension(:, :, intent(in) X, REAL(DP),dimension(3),intent(out) N,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Definition at line 926 of file coordinate_routines.f90.

References COORDINATE_CYLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.23 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
ORIENTATION_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*,
REAL(DP),dimension(:, :, intent(in) *ORIENTATION*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 1624 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_ORIENTATION_SET_PTR(), COORDINATE_SYSTEM_-
USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.24 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_-
SET_PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer *COORDINATE_SYSTEM*,
REAL(DP),dimension(:, :, intent(in) *ORIENTATION*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 1655 of file coordinate_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-
ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.25 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
ORIGIN_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*,
REAL(DP),dimension(:, :, intent(in) *ORIGIN*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 1552 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_ORIGIN_SET_PTR(), COORDINATE_SYSTEM_USER_-
NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-
ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.26 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_-
PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer *COORDINATE_SYSTEM*,
REAL(DP),dimension(:, :, intent(in) *ORIGIN*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 1583 of file coordinate_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-
ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_ORIGIN_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.27 INTEGER(INTG) COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
RADIAL_INTERPOLATION_TYPE_GET (TYPE(COORDINATE_SYSTEM_-
TYPE),intent(in) COORDINATE_SYSTEM)**

Definition at line 1133 of file coordinate_routines.f90.

**6.11.2.28 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_-
INTERPOLATION_TYPE_SET_NUMBER (INTEGER(INTG),intent(in)
USER_NUMBER, INTEGER(INTG),intent(in) RADIAL_INTERPOLATION_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Definition at line 1364 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR(),
COORDINATE_SYSTEM_USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_-
ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.11.2.29 subroutine COORDINATE_ROUTINES::COORDINATE_-
SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR
(TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
INTEGER(INTG),intent(in) RADIAL_INTERPOLATION_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Definition at line 1395 of file coordinate_routines.f90.

References COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_NO_RADIAL_-
INTERPOLATION_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_-
PROLATE_SPHEROIDAL_TYPE, COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE,
COORDINATE_RADIAL_INTERPOLATION_TYPE, COORDINATE_RADIAL_SQUARED_-
INTERPOLATION_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_-
SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and
BASE_ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.11.2.30 INTEGER(INTG) COORDINATE_ROUTINES::COORDINATE_-
SYSTEM_TYPE_GET (TYPE(COORDINATE_SYSTEM_TYPE),intent(in)
COORDINATE_SYSTEM)**

Definition at line 1157 of file coordinate_routines.f90.

6.11.2.31 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*, INTEGER(INTG),intent(in) *TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 1471 of file coordinate_routines.f90.

References COORDINATE_SYSTEM_TYPE_SET_PTR(), COORDINATE_SYSTEM_USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.32 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer *COORDINATE_SYSTEM*, INTEGER(INTG),intent(in) *TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 1502 of file coordinate_routines.f90.

References COORDINATE_CYLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by COORDINATE_SYSTEM_TYPE_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.33 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(COORDINATE_SYSTEM_TYPE),pointer *COORDINATE_SYSTEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 3946 of file coordinate_routines.f90.

References COORDINATE_SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), GLOBAL_COORDINATE_SYSTEM, and KINDS::PTR.

Referenced by COORDINATE_SYSTEM_CREATE_START(), COORDINATE_SYSTEM_DIMENSION_SET_NUMBER(), COORDINATE_SYSTEM_FOCUS_SET_NUMBER(), COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER(), COORDINATE_SYSTEM_ORIGIN_SET_NUMBER(), COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER(), COORDINATE_SYSTEM_TYPE_SET_NUMBER(), REGION_ROUTINES::REGION_CREATE_START(), and REGION_ROUTINES::REGIONS_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.34 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEMS_FINALISE
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 \ast)

Definition at line 3988 of file coordinate_routines.f90.

References COORDINATE_SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_-ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), GLOBAL_COORDINATE_SYSTEM, and KINDS::PTR.

Referenced by CMISS::CMISS_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.35 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 \ast)

Definition at line 4021 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_-SYSTEMS, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), GLOBAL_COORDINATE_SYSTEM, and KINDS::PTR.

Referenced by CMISS::CMISS_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.11.2.36 REAL(DP) COORDINATE_ROUTINES::D2ZX_DP (TYPE(COORDINATE_-
 SYSTEM_TYPE),intent(in) *COORDINATE_SYSTEM*, INTEGER(INTG),intent(in)
 I , INTEGER(INTG),intent(in) J , REAL(DP),dimension(:),intent(in) X ,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*
 [private])

Definition at line 2084 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_-OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_-RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_-ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.37 REAL(DP) COORDINATE_ROUTINES::DXZ_DP (TYPE(COORDINATE_-
 SYSTEM_TYPE),intent(in) *COORDINATE_SYSTEM*, INTEGER(INTG),intent(in)
 I , REAL(DP),dimension(:),intent(in) X , INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 1943 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_-OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_-

RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.2.38 REAL(DP) COORDINATE_ROUTINES::DZX_DP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, INTEGER(INTG),intent(in) I, REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 2353 of file coordinate_routines.f90.

References KINDS::_DP, COORDINATE_CYCLINDRICAL_POLAR_TYPE, COORDINATE_OBLATE_SPHEROIDAL_TYPE, COORDINATE_PROLATE_SPHEROIDAL_TYPE, COORDINATE_RECTANGULAR_CARTESIAN_TYPE, COORDINATE_SPHERICAL_POLAR_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.11.3 Variable Documentation

6.11.3.1 CHARACTER(LEN=21) COORDINATE_ROUTINES::_SYSTEM_TYPE_STRING = & (/ , & , & , & , & /)

Definition at line 106 of file coordinate_routines.f90.

6.11.3.2 CHARACTER(LEN=21) COORDINATE_ROUTINES::COORDINAT

Definition at line 106 of file coordinate_routines.f90.

6.11.3.3 TYPE(COORDINATE_SYSTEMS_TYPE) COORDINATE_ROUTINES::COORDINATE_SYSTEMS

Definition at line 113 of file coordinate_routines.f90.

Referenced by COORDINATE_SYSTEM_CREATE_FINISH(), COORDINATE_SYSTEM_CREATE_START(), COORDINATE_SYSTEM_DESTROY_NUMBER(), COORDINATE_SYSTEM_DESTROY_PTR(), COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_SYSTEMS_FINALISE(), and COORDINATE_SYSTEMS_INITIALISE().

6.11.3.4 TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_ROUTINES::GLOBAL_COORDINATE_SYSTEM

Definition at line 115 of file coordinate_routines.f90.

Referenced by COORDINATE_SYSTEM_USER_NUMBER_FIND(), COORDINATE_SYSTEMS_FINALISE(), and COORDINATE_SYSTEMS_INITIALISE().

6.12 DOMAIN_MAPPINGS Namespace Reference

This module handles all domain mappings routines.

Functions

- subroutine `DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE` (`ADJACENT_DOMAIN, ERR, ERROR,*`)
Finalises the adjacent domain and deallocates all memory for a domain mapping.
- subroutine `DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE` (`ADJACENT_DOMAIN, ERR, ERROR,*`)
Initialise the adjacent domain for a domain mapping.
- subroutine `DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE` (`DOMAIN_MAPPING, ERR, ERROR,*`)
Calculates the domain mappings local map from a domain mappings global map.
- subroutine `DOMAIN_MAPPINGS_MAPPING_FINALISE` (`DOMAIN_MAPPING, ERR, ERROR,*`)
Finalises the mapping for a domain mappings mapping and deallocates all memory.
- subroutine `DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE` (`MAPPING_GLOBAL_MAP, ERR, ERROR,*`)
Finalises the global mapping in the given domain mappings.
- subroutine `DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE` (`MAPPING_GLOBAL_MAP, ERR, ERROR,*`)
Finalises the global mapping in the given domain mappings.
- subroutine `DOMAIN_MAPPINGS_MAPPING_INITIALISE` (`DOMAIN_MAPPING, NUMBER_OF_DOMAINS, ERR, ERROR,*`)
Initialises the mapping for a domain mappings mapping.

Variables

- INTEGER(INTG), parameter `DOMAIN_LOCAL_INTERNAL` = 1
The domain item is internal to the domain.
- INTEGER(INTG), parameter `DOMAIN_LOCAL_BOUNDARY` = 2
The domain item is on the boundary of the domain.
- INTEGER(INTG), parameter `DOMAIN_LOCAL_GHOST` = 3
The domain item is ghosted from another domain.

6.12.1 Detailed Description

This module handles all domain mappings routines.

6.12.2 Function Documentation

6.12.2.1 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE (TYPE(DOMAIN_ADJACENT_DOMAIN_TYPE) *ADJACENT_DOMAIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises the adjacent domain and deallocates all memory for a domain mapping.

Parameters:

ADJACENT_DOMAIN The adjacent domain to finalise.

ERR The error code

ERROR The error string

Definition at line 86 of file domain_mappings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.12.2.2 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE (TYPE(DOMAIN_ADJACENT_DOMAIN_TYPE) *ADJACENT_DOMAIN*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialise the adjacent domain for a domain mapping.

Parameters:

ADJACENT_DOMAIN The adjacent domain to initialise

ERR The error code

ERROR The error string

Definition at line 114 of file domain_mappings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.12.2.3 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE (TYPE(DOMAIN_MAPPING_TYPE),pointer *DOMAIN_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Calculates the domain mappings local map from a domain mappings global map.

Parameters:

DOMAIN_MAPPING The domain mapping to calculate the local mappings

ERR The error code

ERROR The error string

Definition at line 140 of file domain_mappings.f90.

References COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INTG_TYPE, LISTS::LIST_REMOVE_DUPLICATES(), and KINDS::PTR.

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.2.4 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE (TYPE(DOMAIN_MAPPING_TYPE),pointer *DOMAIN_MAPPING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finalises the mapping for a domain mappings mapping and deallocates all memory.

Parameters:

DOMAIN_MAPPING A pointer to the domain mapping to finalise

ERR The error code

ERROR The error string

Definition at line 410 of file domain_mappings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.2.5 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE (TYPE(DOMAIN_GLOBAL_MAPPING_TYPE) *MAPPING_GLOBAL_MAP*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises the global mapping in the given domain mappings.

Parameters:

MAPPING_GLOBAL_MAP The domain global mapping to finalise

ERR The error code

ERROR Th error string

Definition at line 453 of file domain_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.12.2.6 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE (`TYPE(DOMAIN_GLOBAL_MAPPING_TYPE)`
`MAPPING_GLOBAL_MAP`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`)

Finalises the global mapping in the given domain mappings.

Parameters:

`MAPPING_GLOBAL_MAP` The domain global mapping to initialise
`ERR` The error code
`ERROR` The error string

Definition at line 480 of file domain_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, and
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.12.2.7 subroutine DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_INITIALISE (`TYPE(DOMAIN_MAPPING_TYPE),pointer DOMAIN_MAPPING,`
`INTEGER(INTG),intent(in) NUMBER_OF_DOMAINS`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Initialises the mapping for a domain mappings mapping.

Parameters:

`DOMAIN_MAPPING` A pointer to the domain mapping to initialise the mappings for
`NUMBER_OF_DOMAINS` The number of domains
`ERR` The error code
`ERROR` The error string

Definition at line 505 of file domain_mappings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, `MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE()`, and
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.13 ELASTICITY_ROUTINES Namespace Reference

This module handles all elasticity class routines.

6.13.1 Detailed Description

This module handles all elasticity class routines.

6.14 ELECTROMECHANICS_ROUTINES Namespace Reference

This module handles all electromechanics class routines.

6.14.1 Detailed Description

This module handles all electromechanics class routines.

6.15 EQUATIONS_SET_CONSTANTS Namespace Reference

This module defines all constants shared across equations set routines.

Variables

- INTEGER(INTG), parameter `EQUATIONS_SET_NO_CLASS` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_ELASTICITY_CLASS` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_FLUID_MECHANICS_CLASS` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_ELECTROMAGNETICS_CLASS` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_CLASSICAL_FIELD_CLASS` = 4
- INTEGER(INTG), parameter `EQUATIONS_SET_MODAL_CLASS` = 5
- INTEGER(INTG), parameter `EQUATIONS_SET_FITTING_CLASS` = 6
- INTEGER(INTG), parameter `EQUATIONS_SET_OPTIMISATION_CLASS` = 7
- INTEGER(INTG), parameter `EQUATIONS_SET_NO_TYPE` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_LINEAR_ELASTICITY_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SETFINITE_ELASTICITY_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_STOKES_FLUID_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_NAVIER_STOKES_FLUID_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_ELECTROSTATIC_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_MAGNETOSTATIC_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_MAXWELLS_EQUATIONS_TYPE` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_LAPLACE_EQUATION_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_POISSON_EQUATION_TYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_HELMHOLTZ_EQUATION_TYPE` = 3
- INTEGER(INTG), parameter `EQUATIONS_SET_WAVE_EQUATION_TYPE` = 4
- INTEGER(INTG), parameter `EQUATIONS_SET_DIFFUSION_EQUATION_TYPE` = 5
- INTEGER(INTG), parameter `EQUATIONS_SET_ADVECTION_DIFFUSION_EQUATION_TYPE` = 6
- INTEGER(INTG), parameter `EQUATIONS_SETREACTION_DIFFUSION_EQUATION_TYPE` = 7
- INTEGER(INTG), parameter `EQUATIONS_SET_BIHAMONIC_EQUATION_TYPE` = 8
- INTEGER(INTG), parameter `EQUATIONS_SET_LINEAR_ELASTIC_MODAL_TYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_NO_SUBTYPE` = 0
- INTEGER(INTG), parameter `EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE` = 1
- INTEGER(INTG), parameter `EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE` = 2
- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_INITIAL_TYPE` = 1

Initial setup.

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_GEOMETRY_TYPE` = 2

Geometry setup.

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_DEPENDENT_TYPE` = 3

Dependent variables.

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_MATERIALS_TYPE` = 4

Materials setup.

- INTEGER(INTG), parameter `EQUATIONS_SET_SETUP_SOURCE_TYPE` = 5

Source setup.

- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_SOURCE_MATERIALS_TYPE** = 6
Source materials setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_ANALYTIC_TYPE** = 7
Analytic setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE** = 8
Fixed conditions.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_EQUATIONS_TYPE** = 9
Equations setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_FINAL_TYPE** = 9
Final setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_START_ACTION** = 1
Start setup action.
- INTEGER(INTG), parameter **EQUATIONS_SET_SETUP_FINISH_ACTION** = 2
Finish setup action.
- INTEGER(INTG), parameter **EQUATIONS_SET_NOT_FIXED** = 0
The dof is not fixed.
- INTEGER(INTG), parameter **EQUATIONS_SET_FIXED_BOUNDARY_CONDITION** = 1
The dof is fixed as a boundary condition.
- INTEGER(INTG), parameter **EQUATIONS_SET_MIXED_BOUNDARY_CONDITION** = 2
The dof is set as a mixed boundary condition.
- INTEGER(INTG), parameter **NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES** = 3
The number of problem linearity types defined.
- INTEGER(INTG), parameter **EQUATIONS_SET_LINEAR** = 1
The problem is linear.
- INTEGER(INTG), parameter **EQUATIONS_SET_NONLINEAR** = 2
The problem is non-linear.
- INTEGER(INTG), parameter **EQUATIONS_SET_NONLINEAR_BCS** = 3
The problem has non-linear boundary conditions.
- INTEGER(INTG), parameter **NUMBER_OF_EQUATIONS_SET_TIME_TYPES** = 3
The number of problem time dependence types defined.
- INTEGER(INTG), parameter **EQUATIONS_SET_STATIC** = 1
The problem is static and has no time dependence.

- INTEGER(INTG), parameter **EQUATIONS_SET_DYNAMIC** = 2
The problem is dynamic.
- INTEGER(INTG), parameter **EQUATIONS_SET_QUASISTATIC** = 3
The problem is quasi-static.
- INTEGER(INTG), parameter **NUMBER_OF_EQUATIONS_SET SOLUTION_METHODS** = 6
The number of solution methods defined.
- INTEGER(INTG), parameter **EQUATIONS_SET_FEM SOLUTION_METHOD** = 1
Finite Element Method solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_BEM SOLUTION_METHOD** = 2
Boundary Element Method solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_FD SOLUTION_METHOD** = 3
Finite Difference solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_FV SOLUTION_METHOD** = 4
Finite Volume solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_GFEM SOLUTION_METHOD** = 5
Grid-based Finite Element Method solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_GFV SOLUTION_METHOD** = 6
Grid-based Finite Volume solution method.
- INTEGER(INTG), parameter **EQUATIONS_SET_NO_OUTPUT** = 0
No output.
- INTEGER(INTG), parameter **EQUATIONS_SET_TIMING_OUTPUT** = 1
Timing information output.
- INTEGER(INTG), parameter **EQUATIONS_SET_MATRIX_OUTPUT** = 2
All below and equation matrices output.
- INTEGER(INTG), parameter **EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT** = 3
All below and Element matrices output.
- INTEGER(INTG), parameter **EQUATIONS_SET_SPARSE_MATRICES** = 1
Use sparse matrices for the equations set.
- INTEGER(INTG), parameter **EQUATIONS_SET_FULL_MATRICES** = 2
Use fully populated matrices for the equations set.

6.15.1 Detailed Description

This module defines all constants shared across equations set routines.

6.15.2 Variable Documentation

6.15.2.1 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- ADVECTION_DIFFUSION_EQUATION_TYPE = 6

Definition at line 80 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_-ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.15.2.2 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- BIHARMONIC_EQUATION_TYPE = 8

Definition at line 82 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_-ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.15.2.3 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- CLASSICAL_FIELD_CLASS = 4

Definition at line 56 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQ(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP().

6.15.2.4 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- DIFFUSION_EQUATION_TYPE = 5

Definition at line 79 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_-ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.15.2.5 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- ELASTICITY_CLASS = 1

Definition at line 53 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQ(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP().

6.15.2.6 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_- ELECTROMAGNETICS_CLASS = 3

Definition at line 55 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQ(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SETSETUP().

6.15.2.7 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-ELECTROSTATIC_TYPE = 1

Definition at line 71 of file equations_set_constants.f90.

6.15.2.8 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-FINITE_ELASTICITY_TYPE = 2

Definition at line 66 of file equations_set_constants.f90.

6.15.2.9 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-FITTING_CLASS = 6

Definition at line 59 of file equations_set_constants.f90.

6.15.2.10 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-FLUID_MECHANICS_CLASS = 2

Definition at line 54 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQ(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SETSETUP().

6.15.2.11 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-GENERALISED_LAPLACE_SUBTYPE = 2

Definition at line 94 of file equations_set_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET().

6.15.2.12 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET-HELMHOLTZ_EQUATION_TYPE = 3

Definition at line 77 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

**6.15.2.13 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
LAPLACE_EQUATION_TYPE = 1**

Definition at line 75 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP().

**6.15.2.14 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
LINEAR_ELASTIC_MODAL_TYPE = 1**

Definition at line 84 of file equations_set_constants.f90.

**6.15.2.15 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
LINEAR_ELASTICITY_TYPE = 1**

Definition at line 65 of file equations_set_constants.f90.

**6.15.2.16 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
MAGNETOSTATIC_TYPE = 2**

Definition at line 72 of file equations_set_constants.f90.

**6.15.2.17 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
MAXWELLS_EQUATIONS_TYPE = 3**

Definition at line 73 of file equations_set_constants.f90.

**6.15.2.18 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
MODAL_CLASS = 5**

Definition at line 58 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQ(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP().

**6.15.2.19 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NAVIER_STOKES_FLUID_TYPE = 2**

Definition at line 69 of file equations_set_constants.f90.

**6.15.2.20 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
NO_CLASS = 0**

Definition at line 51 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_INITIALISE().

6.15.2.21 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_SUBTYPE = 0

Definition at line 87 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_INITIALISE().

6.15.2.22 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_TYPE = 0

Definition at line 63 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_INITIALISE().

6.15.2.23 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_OPTIMISATION_CLASS = 7

Definition at line 60 of file equations_set_constants.f90.

6.15.2.24 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE = 2

Definition at line 76 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.15.2.25 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SETREACTION_DIFFUSION_EQUATION_TYPE = 7

Definition at line 81 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.15.2.26 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SETSTANDARD_LAPLACE_SUBTYPE = 1

Definition at line 93 of file equations_set_constants.f90.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP(), LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATION_PROBLEM_STANDARD_SETUP().

**6.15.2.27 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
STOKES_FLUID_TYPE = 1**

Definition at line 68 of file equations_set_constants.f90.

**6.15.2.28 INTEGER(INTG),parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
WAVE_EQUATION_TYPE = 4**

Definition at line 78 of file equations_set_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CL(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

6.16 EQUATIONS_SET_ROUTINES Namespace Reference

This module handles all equations set routines.

Classes

- interface [EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF](#)
- interface [EQUATIONS_SET_SPECIFICATION_SET](#)

Functions

- subroutine [EQUATIONS_SET_ANALYTIC_CREATE_FINISH](#) (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of a analytic solution for equations set.
- subroutine [EQUATIONS_SET_ANALYTIC_CREATE_START](#) (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of a analytic solution for a equations set.
- subroutine [EQUATIONS_SET_ANALYTIC_DESTROY](#) (EQUATIONS_SET, ERR, ERROR,*)

Destroy the analytic solution for an equations set.
- subroutine [EQUATIONS_SET_ANALYTIC_FINALISE](#) (EQUATIONS_SET_ANALYTIC, ERR, ERROR,*)

Finalise the analytic solution for an equations set and deallocate all memory.
- subroutine [EQUATIONS_SET_ANALYTIC_INITIALISE](#) (EQUATIONS_SET, ERR, ERROR,*)

Initialises the analytic solution for an equations set.
- subroutine [EQUATIONS_SET_ASSEMBLE](#) (EQUATIONS_SET, ERR, ERROR,*)

Assembles the equations for an equations set.
- subroutine [EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM](#) (EQUATIONS_SET, ERR, ERROR,*)

Assembles the equations stiffness matrix and rhs for a linear static equations set using the finite element method.
- subroutine [EQUATIONS_SET_BACKSUBSTITUTE](#) (EQUATIONS_SET, ERR, ERROR,*)

Backsubstitutes with an equations set to calculate unknown right hand side vectors.
- subroutine [EQUATIONS_SET_CREATE_FINISH](#) (EQUATIONS_SET, ERR, ERROR,*)

Finishes the process of creating an equation set on a region.
- subroutine [EQUATIONS_SET_CREATE_START](#) (USER_NUMBER, REGION, GEOM_FIBRE_FIELD, EQUATIONS_SET, ERR, ERROR,*)

Starts the process of creating an equations set defined by USER_NUMBER in the region identified by REGION.
- subroutine [EQUATIONS_SET_DESTROY](#) (USER_NUMBER, REGION, ERR, ERROR,*)

Destroys an equations set identified by a user number on the give region and deallocates all memory.

- subroutine [EQUATIONS_SET_FINALISE](#) (EQUATIONS_SET, ERR, ERROR,*)

Finalise the equations set and deallocate all memory.
- subroutine [EQUATIONS_SETFINITEELEMENTCALCULATE](#) (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matrices and rhs vector for the given element number for a finite element equations set.
- subroutine [EQUATIONSINTERPOLATIONFINALISE](#) (EQUATIONS_INTERPOLATION, ERR, ERROR,*)

Finalises the interpolation information for equations and deallocates all memory.
- subroutine [EQUATIONSINTERPOLATIONINITIALISE](#) (EQUATIONS, ERR, ERROR,*)

Initialises the interpolation information for equations.
- subroutine [EQUATIONSSETINITIALISE](#) (EQUATIONS_SET, ERR, ERROR,*)

Initialises an equations set.
- subroutine [EQUATIONSSETFIXEDCONDITIONSAPPLY](#) (EQUATIONS_SET, ERR, ERROR,*)

Applies the fixed conditions in an equation set to the dependent field in an equations set.
- subroutine [EQUATIONSSETFIXEDCONDITIONSCREATEFINISH](#) (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of fixed conditions for an equation set.
- subroutine [EQUATIONSSETFIXEDCONDITIONSCREATESTART](#) (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of fixed conditions for a problem.
- subroutine [EQUATIONSSETFIXEDCONDITIONSDESTROY](#) (EQUATIONS_SET, ERR, ERROR,*)

Destroy the fixed conditions for an equations set and deallocate all memory.
- subroutine [EQUATIONSSETFIXEDCONDITIONSFINALISE](#) (EQUATIONS_SET_FIXED_CONDITIONS, ERR, ERROR,*)

Finalise the fixed conditions for an equations set and deallocate all memory.
- subroutine [EQUATIONSSETFIXEDCONDITIONSINITIALISE](#) (EQUATIONS_SET, ERR, ERROR,*)

Initialises the fixed conditions for a problem.
- subroutine [EQUATIONSSETFIXEDCONDITIONSSETDOFS](#) (EQUATIONS_SET, DOF_INDICES, CONDITIONS, VALUES, ERR, ERROR,*)

Sets fixed conditions for the equations set on the specified dofs.
- subroutine [EQUATIONSSETFIXEDCONDITIONSSETDOF1](#) (EQUATIONS_SET, DOF_INDEX, CONDITION, VALUE, ERR, ERROR,*)

Sets a fixed condition for the equation set on the specified dof.

- subroutine **EQUATIONS_SET_GEOMETRY_FINALISE** (EQUATIONS_SET_GEOMETRY, ERR, ERROR,*)

Finalise the geometry for an equations set.
- subroutine **EQUATIONS_SET_GEOMETRY_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the geometry for an equation set.
- subroutine **EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE** (EQUATIONS_LINEAR_DATA, ERR, ERROR,*)

Finalise the equations set linear data and deallocate all memory.
- subroutine **EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE** (EQUATIONS, ERR, ERROR,*)

Initialises the linear data information for an equations.
- subroutine **EQUATIONS_SET_MATERIALS_COMPONENT_INTERPOLATION_SET** (EQUATIONS_SET, COMPONENT_NUMBER, INTERPOLATION_TYPE, ERR, ERROR,*)

Sets/changes the field component interpolation for a materials field of a problem.
- subroutine **EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_SET** (EQUATIONS_SET, COMPONENT_NUMBER, MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Sets/changes the field component mesh component for a materials field of a problem.
- subroutine **EQUATIONS_SET_MATERIALS_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of materials for an equations set.
- subroutine **EQUATIONS_SET_MATERIALS_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of materials for a problem.
- subroutine **EQUATIONS_SET_MATERIALS_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)

Destroy the materials for an equations set.
- subroutine **EQUATIONS_SET_MATERIALS_FINALISE** (EQUATIONS_SET_MATERIALS, ERR, ERROR,*)

Finalise the materials for an equations set.
- subroutine **EQUATIONS_SET_MATERIALS_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the materials for an equations set.
- subroutine **EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET** (EQUATIONS_SET, MATERIAL_FIELD, ERR, ERROR,*)

Returns a pointer to the material field of the materials for a problem.
- subroutine **EQUATIONS_SET_MATERIALS_SCALING_SET** (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a materials field of an equations set.

- subroutine [EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE](#) (EQUATIONS_SET, EQUATIONS_NONLINEAR_DATA, ERR, ERROR,*)

Finalise the equations set nonlinear data and deallocate all memory.

- subroutine [EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_INITIALISE](#) (EQUATIONS_SET, EQUATIONS_NONLINEAR_DATA, ERR, ERROR,*)

Initialises the nonlinear data information for an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_](#) (EQUATIONS_SET, VARIABLE_NUMBER, COMPONENT_NUMBER, &MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Sets/changes the field component mesh component for a dependent field of an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_CREATE_FINISH](#) (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of a dependent variables for an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_CREATE_START](#) (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of dependent variables for an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_DEPENDENT_FIELD_GET](#) (EQUATIONS_SET, DEPENDENT_FIELD, ERR, ERROR,*)

Returns a pointer to the dependent field of the dependent variables for an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_DESTROY](#) (EQUATIONS_SET, ERR, ERROR,*)

Destroy the dependent variables for an equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_FINALISE](#) (EQUATIONS_SET_DEPENDENT, ERR, ERROR,*)

Finalises the dependent variables for an equation set and deallocates all memory.

- subroutine [EQUATIONS_SET_DEPENDENT_INITIALISE](#) (EQUATIONS_SET, ERR, ERROR,*)

Initialises the dependent variables for a equations set.

- subroutine [EQUATIONS_SET_DEPENDENT_SCALING_SET](#) (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a dependent field of an equations set.

- subroutine [EQUATIONS_SET_SETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the specifics for an equation set.

- subroutine [EQUATIONS_SET_EQUATIONS_CREATE_FINISH](#) (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of equations for the equations set.

- subroutine [EQUATIONS_SET_EQUATIONS_CREATE_START](#) (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of equations for the equation set.

- subroutine **EQUATIONS_SET_EQUATIONS_FINALISE** (EQUATIONS, ERR, ERROR,*)

Finalise the equations and deallocate all memory.
- subroutine **EQUATIONS_SET_EQUATIONS_SPARSITY_TYPE_SET** (EQUATIONS_SET, SPARSITY_TYPE, ERR, ERROR,*)

Sets/changes the sparsity type for the equations set.
- subroutine **EQUATIONS_SET_EQUATIONS_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the equations for an equations set.
- subroutine **EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET** (EQUATIONS_SET, OUTPUT_TYPE, ERR, ERROR,*)

Sets/changes the output type for the equations set.
- subroutine **EQUATIONS_SET_SOURCE_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of a source for an equation set.
- subroutine **EQUATIONS_SET_SOURCE_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of a source for an equations set.
- subroutine **EQUATIONS_SET_SOURCE_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)

Destroy the source for an equations set.
- subroutine **EQUATIONS_SET_SOURCE_FINALISE** (EQUATIONS_SET_SOURCE, ERR, ERROR,*)

Finalise the source for a equations set and deallocate all memory.
- subroutine **EQUATIONS_SET_SOURCE_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the source for an equations set.
- subroutine **EQUATIONS_SET_SOURCE_SCALING_SET** (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a source field of an equations set.
- subroutine **EQUATIONS_SET_SPECIFICAT** (USER_NUMBER, REGION, EQUATIONS_SET_CLASS, EQUATIONS_SET_TYPE_, &EQUATIONS_SET_SUBTYPE, ERR, ERROR,*)

Sets/changes the equation set specification i.e., equation set class, type and subtype for an equation set identified by a user number.
- subroutine **EQ** (EQUATIONS_SET, EQUATIONS_SET_CLASS, EQUATIONS_SET_TYPE_, EQUATIONS_SET_SUBTYPE, &ERR, ERROR,*)

Sets/changes the equations set specification i.e., equations set class, type and subtype for a equations set identified by a pointer.
- subroutine **EQUATIONS_SET_EQUATIONS_TIME_DATA_FINALISE** (TIME_DATA, ERR, ERROR,*)

Finalises the time data information for an equations and deallocates all memory.

- subroutine **EQUATIONS_SET_EQUATIONS_TIME_DATA_INITIALISE** (EQUATIONS, ERR, ERROR,*)

Initialises the time data information for an equations.

- subroutine **EQUATIONS_SET_USER_NUMBER_FIND** (USER_NUMBER, REGION, EQUATIONS_SET, ERR, ERROR,*)

Finds and returns in EQUATIONS_SET a pointer to the equations set identified by USER_NUMBER in the given REGION. If no equations set with that USER_NUMBER exists EQUATIONS_SET is left nullified.

- subroutine **EQUATIONS_SETS_FINALISE** (REGION, ERR, ERROR,*)

Finalises all equations sets on a region and deallocate all memory.

- subroutine **EQUATIONS_SETS_INITIALISE** (REGION, ERR, ERROR,*)

Initialises all equations sets on a region.

6.16.1 Detailed Description

This module handles all equations set routines.

6.16.2 Function Documentation

- 6.16.2.1 subroutine EQUATIONS_SET_ROUTINES::EQ** (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *EQUATIONS_SET_CLASS*, INTEGER(INTG),intent(in) *EQUATIONS_SET_TYPE_*, INTEGER(INTG),intent(in) *EQUATIONS_SET_SUBTYPE*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *) [private]

Sets/changes the equations set specification i.e., equations set class, type and subtype for a equations set identified by a pointer.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the specification for

EQUATIONS_SET_CLASS The equations set class to set

EQUATIONS_SET_TYPE_ The equations set type to set

EQUATIONS_SET_SUBTYPE The equations set subtype to set

Definition at line 3094 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELASTICITY_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELECTROMAGNETICS_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FLUID_MECHANICS_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MODAL_CLASS, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.2 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_FINALISE (TYPE(EQUATIONS_INTERPOLATION_TYPE),pointer EQUATIONS_INTERPOLATION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalises the interpolation information for equations and deallocates all memory.

Parameters:

EQUATIONS_INTERPOLATION A pointer to the equations interpolation to finalise
ERR The error code
ERROR The error string

Definition at line 1108 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_FINALISE(), and FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE().

Here is the call graph for this function:

6.16.2.3 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE (TYPE(EQUATIONS_TYPE),pointer EQUATIONS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Initialises the interpolation information for equations.

Parameters:

EQUATIONS The pointer to the equations to initialise the interpolation for
ERR The error code
ERROR The error string

Definition at line 1146 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), and FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE.

Here is the call graph for this function:

6.16.2.4 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_FINISH (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finish the creation of a analytic solution for equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to create the analytic for.

ERR The error code

ERROR The error string

Definition at line 122 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.5 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Start the creation of a analytic solution for a equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of an analytic for.

ERR The error code

ERROR The error string

Definition at line 162 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.6 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_DESTROY (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Destroy the analytic solution for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to destroy the analytic solutins for.

ERR The error code

ERROR The error string

Definition at line 200 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.7 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_FINALISE (TYPE(EQUATIONS_SET_ANALYTIC_TYPE),pointer EQUATIONS_SET_ANALYTIC, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalise the analytic solution for an equations set and deallocate all memory.

Parameters:

EQUATIONS_SET_ANALYTIC A pointer to the equations set analytic to finalise

ERR The error code

ERROR The error string

Definition at line 232 of file equations_set_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.16.2.8 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_INITIALISE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Initialises the analytic solution for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the analytic solution for.

ERR The error code

ERROR The error string

Definition at line 258 of file equations_set_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.16.2.9 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Assembles the equations for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the analytic solution for.

ERR The error code

ERROR The error string

Definition at line 296 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR_BCS`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `PROBLEM_ROUTINES::PROBLEM SOLUTION SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.16.2.10 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Assembles the equations stiffness matrix and rhs for a linear static equations set using the finite element method.

Parameters:

EQUATIONS_SET A pointer to the equations set to assemble the equations for
ERR The error code
ERROR The error string

Definition at line 371 of file equations_set_routines.f90.

References `KINDS::_DP`, `KINDS::_SP`, `TIMER::CPU_TIMER()`, `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_TIMING_OUTPUT`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`, `BASE_ROUTINES::GENERAL_OUTPUT_TYPE`, and `KINDS::PTR`.

Here is the call graph for this function:

6.16.2.11 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Backsubstitutes with an equations set to calculate unknown right hand side vectors.

Parameters:

EQUATIONS_SET A pointer to the equations set to backsubstitute

ERR The error code

ERROR The error string

Definition at line 540 of file equations_set_routines.f90.

References KINDS::DP, BASE_ROUTINES::ENTERS(), EQUATIONS_SET_-
 CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION, EQUATIONS_SET_-
 CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION, EQUATIONS_SET_-
 CONSTANTS::EQUATIONS_SET_NOT_FIXED, BASE_ROUTINES::ERRORS(), BASE_-
 ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_VALUES_SET_TYPE, and KINDS::PTR.

Referenced by PROBLEM_ROUTINES::PROBLEM SOLUTION SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.12 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH
 (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)**

Finishes the process of creating an equation set on a region.

Parameters:

EQUATIONS_SET A pointer to the equations set to finish creating

ERR The error code

ERROR The error string

Definition at line 752 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.13 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START
 (INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION,
 TYPE(FIELD_TYPE),pointer GEOM_FIBRE_FIELD,
 TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *) [private]**

Starts the process of creating an equations set defined by USER_NUMBER in the region identified by REGION.

Parameters:

USER_NUMBER The user number of the equations set

REGION A pointer to the region to create the equations set on

GEOM_FIBRE_FIELD A pointer to the either the geometry or, in appropriate, the fibre field for the equation set

EQUATIONS_SET On return, a pointer to the equations set

ERR The error code

ERROR The error string

Definition at line 790 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_FIBRE_TYPE, FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE, and KINDS::PTR.

Here is the call graph for this function:

6.16.2.14 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_(TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in) COMPONENT_NUMBER, &,intent(in) MESH_COMPONENT_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets/changes the field component mesh component for a dependent field of an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the dependent field component mesh component

VARIABLE_NUMBER The dependent field variable number to set this should be variable type???

COMPONENT_NUMBER The dependent field component number to set

ERR The error code

ERROR The error string

Definition at line 2277 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.15 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_FINISH (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Finish the creation of a dependent variables for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to finish the creation of

ERR The error code

ERROR The error string

Definition at line 2314 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.16 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Start the creation of dependent variables for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of a dependent field on
ERR The error code
ERROR The error string

Definition at line 2350 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.17 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_DEPENDENT_FIELD_GET (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, TYPE(FIELD_TYPE),pointer *DEPENDENT_FIELD*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Returns a pointer to the dependent field of the dependent variables for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equation set to get the dependent field for
DEPENDENT_FIELD On return, a pointer to the dependent field for the equations set. Must not be associated on entry.
ERR The error code
ERROR The error string

Definition at line 2383 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.18 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_DESTROY (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Destroy the dependent variables for an equations sety.

Parameters:

EQUATIONS_SET The pointer to the equations set to destroy

ERR The error code

ERROR The error string

Definition at line 2420 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.19 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_FINALISE (TYPE(EQUATIONS_SET_DEPENDENT_TYPE)
EQUATIONS_SET_DEPENDENT, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises the dependent variables for an equation set and deallocates all memory.

Parameters:

EQUATIONS_SET_DEPENDENT The pointer to the equations set

ERR The error code

ERROR The error string

Definition at line 2448 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and FIELD_ROUTINES::FIELD_DESTROY().

Here is the call graph for this function:

6.16.2.20 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_INITIALISE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]

Initialises the dependent variables for a equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the dependent field for

ERR The error code

ERROR The error string

Definition at line 2473 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.21 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_SCALING_SET (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *SCALING_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the field scaling for a dependent field of an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to the dependent field scaling for.

SCALING_TYPE The scaling type to set.

ERR The error code

ERROR The error string

Definition at line 2503 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.22 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DESTROY (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Destroys an equations set identified by a user number on the give region and deallocates all memory.

Parameters:

USER_NUMBER The user number of the equations set to destroy

REGION The region of the equations set to destroy

ERR The error code

ERROR The error string

Definition at line 895 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

6.16.2.23 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finish the creation of equations for the equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to finish the creation of the equations for.

ERR The error code

ERROR The error string

Definition at line 2582 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUATIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.24 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Start the creation of equations for the equation set.

Todo

Should this return a pointer to the equations???

Parameters:

EQUATIONS_SET A pointer to the equations set to create equations for

ERR The error code

ERROR The error string

Definition at line 2624 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUATIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.25 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_FINALISE (TYPE(EQUATIONS_TYPE),pointer EQUATIONS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalise the equations and deallocate all memory.

Parameters:

EQUATIONS A pointer to the equations to finalise

ERR The error code

ERROR The error string

Definition at line 2660 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.26 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]

Initialises the equations for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the equations for
ERR The error code
ERROR The error string

Definition at line 2740 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_OUTPUT, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.27 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_FINALISE (TYPE(EQUATIONS_LINEAR_DATA_TYPE),pointer *EQUATIONS_LINEAR_DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalise the equations set linear data and deallocate all memory.

Parameters:

EQUATIONS_LINEAR_DATA A pointer to the equations linear data to finalise.
ERR The error code
ERROR The error string

Definition at line 1817 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.28 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_LINEAR_DATA_INITIALISE (TYPE(EQUATIONS_TYPE),pointer *EQUATIONS*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]

Initialises the linear data information for an equations.

Parameters:

EQUATIONS The pointer to the equations to initialise the linear data for
ERR The error code
ERROR The error string

Definition at line 1843 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.29 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE (TYPE(EQUATIONS_NONLINEAR_DATA_TYPE),pointer EQUATIONS_NONLINEAR_DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalise the equations set nonlinear data and deallocate all memory.

Parameters:

EQUATIONS_NONLINEAR_DATA A pointer to the equations nonlinear data to finalise.

ERR The error code

ERROR The error string

Definition at line 2214 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.30 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_INITIALISE (TYPE(EQUATIONS_TYPE),pointer EQUATIONS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Initialises the nonlinear data information for an equations.

Parameters:

EQUATIONS The pointer to the equations to initialise the nonlinear data for

ERR The error code

ERROR The error string

Definition at line 2240 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.31 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) OUTPUT_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the output type for the equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the output type for

OUTPUT_TYPE The output type to set

See also:

EQUATIONS_ROUTINES_OutputTypes,EQUATIONS_ROUTINES

ERR The error code

ERROR The error string

Definition at line 2783 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_OUTPUT, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_TIMING_OUTPUT, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.32 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_SPARSITY_TYPE_SET (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) SPARSITY_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the sparsity type for the equations set.

Parameters:

EQUATIONS_SET A pointer to the equations to set the sparsity type for

SPARSITY_TYPE The sparsity type to set

See also:

EQUATIONS_ROUTINES_SparsityTypes,PROBLEM_ROUTINES

ERR The error code

ERROR The error string

Definition at line 2691 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.33 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_DATA_FINALISE (TYPE(EQUATIONS_TIME_DATA_TYPE),pointer TIME_DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalises the time data information for an equations and deallocates all memory.

Parameters:

TIME_DATA A pointer to the equations set time data to finalise

ERR The error code

ERROR The error string

Definition at line 3145 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.34 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_-
TIME_DATA_INITIALISE (TYPE(EQUATIONS_TYPE),pointer EQUATIONS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast$ ) [private]
```

Initialises the time data information for an equations.

Parameters:

EQUATIONS The pointer to the equations to initialise the time data for

ERR The error code

ERROR The error string

Definition at line 3171 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.35 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FINALISE
(TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast$ ) [private]
```

Finalise the equations set and deallocate all memory.

Parameters:

EQUATIONS_SET A pointer to the equations set to finalise.

ERR The error code

ERROR The error string

Definition at line 979 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.36 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_-ELEMENT_CALCULATE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) ELEMENT_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Calculates the element stiffness matrices and rhs vector for the given element number for a finite element equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set
ELEMENT_NUMBER The element number to calcualte
ERR The error code
ERROR The error string

Definition at line 1012 of file equations_set_routines.f90.

References CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE(), BASE_ROUTINES::ENTERS(), SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS, SET_CONSTANTS::EQUATIONS_SET_ELASTICITY_CLASS, CONSTANTS::EQUATIONS_SET_ELECTROMAGNETICS_CLASS, CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT, CONSTANTS::EQUATIONS_SET_FLUID_MECHANICS_CLASS, CONSTANTS::EQUATIONS_SET_MODAL_CLASS, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), BASE_ROUTINES::GENERAL_OUTPUT_TYPE, KINDS::PTR, and INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES.

Here is the call graph for this function:

6.16.2.37 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-CONDITIONS_APPLY (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Applies the fixed conditions in an equation set to the dependent field in an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to apply the fixed conditions for.
ERR The error code
ERROR The error string

Definition at line 1288 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE, and FIELD_ROUTINES::FIELD_VALUES_SET_TYPE.

Referenced by PROBLEM_ROUTINES::PROBLEM SOLUTION SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.38 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-
CONDITIONS_CREATE_FINISH (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_-
STRING),intent(out) ERROR, *)**

Finish the creation of fixed conditions for an equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set to create the fixed conditions for.

ERR The error code

ERROR The error string

Definition at line 1336 of file equations_set_routines.f90.

References COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET(), BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and CMISS_MPI::MPI_ERROR_CHECK().

Here is the call graph for this function:

**6.16.2.39 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-
CONDITIONS_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_-
STRING),intent(out) ERROR, *)**

Start the creation of fixed conditions for a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of the fixed conditions for

ERR The error code

ERROR The error string

Definition at line 1413 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

**6.16.2.40 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-
CONDITIONS_DESTROY (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_-
STRING),intent(out) ERROR, *)**

Destroy the fixed conditions for an equations set and deallocate all memory.

Parameters:

EQUATIONS_SET A pointer to the equations set to destroy the fixed conditions for
ERR The error code
ERROR The error string

Definition at line 1475 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.41 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_FINALISE (`TYPE(EQUATIONS_SET_FIXED_CONDITIONS_TYPE)`,pointer **EQUATIONS_SET_FIXED_CONDITIONS**,
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`,
`*) [private]`

Finalise the fixed conditions for an equations set and deallocate all memory.

Parameters:

EQUATIONS_SET_FIXED_CONDITIONS A pointer to the equations set fixed conditions to finalise.
ERR The error code
ERROR The error string

Definition at line 1507 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.42 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE (`TYPE(EQUATIONS_SET_TYPE)`,pointer **EQUATIONS_SET**, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, `*) [private]`

Initialises the fixed conditions for a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the fixed conditions for
ERR The error code
ERROR The error string

Definition at line 1536 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.43 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-
CONDITIONS_SET_DOF1 (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(in) DOF_INDEX,
INTEGER(INTG),intent(in) CONDITION, REAL(DP),intent(in) VALUE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Sets a fixed condition for the equation set on the specified dof.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the fixed condition for
DOF_INDEX The dof index to set the fixed condition at
CONDITION The fixed condition to set
VALUE The value of the fixed condition to set
ERR The error code
ERROR The error string

Definition at line 1678 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_FIXED_BOUNDARY_CONDITION, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_MIXED_BOUNDARY_CONDITION, EQUATIONS_SET_CONSTANTS::EQUATIONS_-
SET_NOT_FIXED, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and FIELD_-
ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE.

Here is the call graph for this function:

**6.16.2.44 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_-
CONDITIONS_SET_DOFS (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),dimension(:,intent(in)
DOF_INDICES, INTEGER(INTG),dimension(:,intent(in) CONDITIONS,
REAL(DP),dimension(:,intent(in) VALUES, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Sets fixed conditions for the equations set on the specified dofs.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the fixed conditions for.
DOF_INDICES DOF_INDICES(i). The dof index for the i'th dof to set the fixed conditions for
CONDITIONS CONDITIONS(i). The fixed condition for the i'th dof.
VALUES VALUES(i). The value of the fixed condition for the i'th dof.
ERR The error code
ERROR The error string

Definition at line 1572 of file equations_set_routines.f90.

References DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST, BASE_ROUTINES::ENTERS(),
EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION,
EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION,
EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED, BASE_ROUTINES::ERRORS(),

BASE_ROUTINES::EXITS(), and FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_-TYPE.

Here is the call graph for this function:

```
6.16.2.45 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_-
GEOMETRY_FINALISE (TYPE(EQUATIONS_SET_GEOMETRY_TYPE)
EQUATIONS_SET_GEOMETRY, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Finalise the geometry for an equations set.

Parameters:

EQUATIONS_SET_GEOMETRY A pointer to the equations set geometry to finalise
ERR The error code
ERROR The error string

Definition at line 1762 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

```
6.16.2.46 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_GEOMETRY_-
INITIALISE (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]
```

Initialises the geometry for an equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the geometry for.
ERR The error code
ERROR The error string

Definition at line 1787 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

```
6.16.2.47 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_INITIALISE
(TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]
```

Initialises an equations set.

Parameters:

EQUATIONS_SET The pointer to the equations set to initialise

ERR The error code

ERROR The error string

Definition at line 1244 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_CLASS`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_TYPE`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.48 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_INTERPOLATION_SET (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *COMPONENT_NUMBER*, INTEGER(INTG),intent(in) *INTERPOLATION_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets/changes the field component interpolation for a materials field of a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the materials component interpolation for

COMPONENT_NUMBER The component of the material field to set the interpolation for

INTERPOLATION_TYPE The interpolation type to set

ERR The error code

ERROR The error string

Definition at line 1880 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`.

Here is the call graph for this function:

6.16.2.49 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_SET (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(in) *COMPONENT_NUMBER*, INTEGER(INTG),intent(in) *MESH_COMPONENT_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the field component mesh component for a materials field of a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the materials field component mesh component for

COMPONENT_NUMBER The component number of the equations set materials field to set the mesh component for

MESH_COMPONENT_NUMBER The mesh component number to set

ERR The error code

ERROR The error string

Definition at line 1919 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`.

Here is the call graph for this function:

6.16.2.50 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_FINISH (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finish the creation of materials for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to finish the creation of the materials field for

ERR The error code

ERROR The error string

Definition at line 1958 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.51 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Start the creation of materials for a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of the materials field for

ERR The error code

ERROR The error string

Definition at line 1998 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.16.2.52 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_-
DESTROY (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)**

Destroy the materials for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to destroy the materials for

ERR The error code

ERROR The error string

Definition at line 2037 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.16.2.53 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_-
FINALISE (TYPE(EQUATIONS_SET_MATERIALS_TYPE),pointer
EQUATIONS_SET_MATERIALS, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Finalise the materials for an equations set.

Parameters:

EQUATIONS_SET_MATERIALS A pointer to the equations set materials to finalise

ERR The error code

ERROR The error string

Definition at line 2069 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and FIELD_ROUTINES::FIELD_DESTROY().

Here is the call graph for this function:

**6.16.2.54 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_-
INITIALISE (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Initialises the materials for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the materials for

ERR The error code

ERROR The error string

Definition at line 2097 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.55 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_-
MATERIAL_FIELD_GET (TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, TYPE(FIELD_TYPE),pointer MATERIAL_FIELD,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]
```

Returns a pointer to the material field of the materials for a problem.

Parameters:

EQUATIONS_SET A pointer to the equations set to get the material field for

MATERIAL_FIELD On return, a pointer to the materials field for the equations set. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 2136 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.16.2.56 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_-
SCALING_SET (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,
INTEGER(INTG),intent(in) SCALING_TYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Sets/changes the field scaling for a materials field of an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the scaling on the material field

SCALING_TYPE The scaling type to set

ERR The error code

ERROR The error string

Definition at line 2177 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.57 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP
 (*TYPE(EQUATIONS_SET_TYPE)*,*pointer EQUATIONS_SET*,
INTEGER(INTG),intent(in) SETUP_TYPE, *INTEGER(INTG),intent(in) ACTION_TYPE*,
INTEGER(INTG),intent(out) ERR, *TYPE(VARYING_STRING),intent(out) ERROR*, *) [private]

Sets up the specifics for an equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set to perform the setup on

SETUP_TYPE The type of setup

See also:

EQUATIONS_ROUTINES_SetupTypes,EQUATIONS_ROUTINES

ACTION_TYPE The setup type action

See also:

EQUATIONS_ROUTINES_SetupActionTypes,EQUATIONS_ROUTINES

ERR The error code

ERROR The error string

Definition at line 2536 of file equations_set_routines.f90.

References CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP(), BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELASTICITY_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELECTROMAGNETICS_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FLUID_MECHANICS_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MODAL_CLASS, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.58 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_FINISH
 (*TYPE(EQUATIONS_SET_TYPE)*,*pointer EQUATIONS_SET*,
INTEGER(INTG),intent(out) ERR, *TYPE(VARYING_STRING),intent(out) ERROR*, *)

Finish the creation of a source for an equation set.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of a souce for

ERR The error code

ERROR The error string

Definition at line 2835 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE, BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.16.2.59 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_START (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Start the creation of a source for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to start the creation of a source for

ERR The error code

ERROR The error string

Definition at line 2875 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.60 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_DESTROY (TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Destroy the source for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to destroy the source for

ERR The error code

ERROR The error string

Definition at line 2913 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.61 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE (TYPE(EQUATIONS_SET_SOURCE_TYPE),pointer *EQUATIONS_SET_SOURCE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalise the source for a equations set and deallocate all memory.

Parameters:

EQUATIONS_SET_SOURCE A pointer to the equations set source to finalise

ERR The error code

ERROR The error string

Definition at line 2945 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_ROUTINES::FIELD_DESTROY()`.

Here is the call graph for this function:

6.16.2.62 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_INITIALISE (`TYPE(EQUATIONS_SET_TYPE)`,pointer *EQUATIONS_SET*,
`INTEGER(INTG),intent(out)` *ERR*, `TYPE(VARYING_STRING),intent(out)` *ERROR*,
`*) [private]`

Initialises the source for an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to initialise the source field for.

ERR The error code

ERROR The error string

Definition at line 2972 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.63 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_SCALING_SET (`TYPE(EQUATIONS_SET_TYPE)`,pointer *EQUATIONS_SET*,
`INTEGER(INTG),intent(in)` *SCALING_TYPE*, `INTEGER(INTG),intent(out)` *ERR*,
`TYPE(VARYING_STRING),intent(out)` *ERROR*, `*)`

Sets/changes the field scaling for a source field of an equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the scaling on the source field

SCALING_TYPE The scaling type to set

ERR The error code

ERROR The error string

Definition at line 3010 of file equations_set_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.16.2.64 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICAT
(INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer
***REGION*, INTEGER(INTG),intent(in) *EQUATIONS_SET_CLASS*,**
INTEGER(INTG),intent(in) *EQUATIONS_SET_TYPE_*, &,intent(in)
***EQUATIONS_SET_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*,**
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets/changes the equation set specification i.e., equation set class, type and subtype for an equation set identified by a user number.

Parameters:

USER_NUMBER The user number of the equations set
REGION A pointer to the region containing the equations set
EQUATIONS_SET_CLASS The equations set class to set
EQUATIONS_SET_TYPE_ The equations set equation type to set
ERR The error code
ERROR The error string

Definition at line 3047 of file equations_set_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.16.2.65 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SET_USER_NUMBER_-
FIND (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer
***REGION*, TYPE(EQUATIONS_SET_TYPE),pointer *EQUATIONS_SET*,**
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
***) [private]**

Finds and returns in **EQUATIONS_SET** a pointer to the equations set identified by **USER_NUMBER** in the given **REGION**. If no equations set with that **USER_NUMBER** exists **EQUATIONS_SET** is left nullified.

Parameters:

USER_NUMBER The user number to find the equation set
REGION The region to find the equations set in
EQUATIONS_SET On return, a pointer to the equations set if an equations set with the specified user number exists in the given region. If no equation set with the specified number exists a NULL pointer is returned. The pointer must not be associated on entry.
ERR The error code
ERROR The error string

Definition at line 3208 of file equations_set_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **KINDS::PTR**.

Here is the call graph for this function:

**6.16.2.66 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE
(TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Finalises all equations sets on a region and deallocates all memory.

Parameters:

REGION A pointer to the region to finalise the problems for.

ERR The error code

ERROR The error string

Definition at line 3258 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Referenced by REGION_ROUTINES::REGION_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.16.2.67 subroutine EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE
(TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Initialises all equations sets on a region.

Parameters:

REGION A pointer to the region to initialise the equations sets for

ERR The error code

ERROR The error string

Definition at line 3293 of file equations_set_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by REGION_ROUTINES::REGION_CREATE_START(), and REGION_-ROUTINES::REGION_SUB_REGION_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.17 F90C Namespace Reference

This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c_c.c](#) module.

Classes

- interface [interface](#)

Functions

- INTEGER(INTG) [CSTRINGLENGTH](#) (CSTRING)
- subroutine [C2FSTRING](#) (CSTRING, FSTRING, ERR, ERROR,*)
- INTEGER(INTG) [FSTRINGLENGTH](#) (FSTRING)
- subroutine [F2CSTRING](#) (CSTRING, FSTRING, ERR, ERROR,*, FSTRINGLEN)

6.17.1 Detailed Description

This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c_c.c](#) module.

6.17.2 Function Documentation

6.17.2.1 subroutine [F90C::C2FSTRING](#) (INTEGER(INTG),dimension(*),intent(in) *CSTRING*, CHARACTER(LEN=*),intent(out) *FSTRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 110 of file f90c_f.f90.

References [CSTRINGLENGTH\(\)](#), [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Referenced by [BINARY_FILE::CLOSE_BINARY_FILE\(\)](#), [TIMER::CPU_TIMER\(\)](#), [BINARY_FILE::INQUIRE_EOF_BINARY_FILE\(\)](#), [BINARY_FILE::OPEN_BINARY_FILE\(\)](#), [BINARY_FILE::READ_BINARY_FILE_CHARACTER\(\)](#), [BINARY_FILE::READ_BINARY_FILE_DP\(\)](#), [BINARY_FILE::READ_BINARY_FILE_DP1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_DPC\(\)](#), [BINARY_FILE::READ_BINARY_FILE_DPC1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_INTG\(\)](#), [BINARY_FILE::READ_BINARY_FILE_LINTG\(\)](#), [BINARY_FILE::READ_BINARY_FILE_LINTG1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_LOGICAL\(\)](#), [BINARY_FILE::READ_BINARY_FILE_LOGICAL1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SINTG\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SINTG1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SP\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SP1\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SPC\(\)](#), [BINARY_FILE::READ_BINARY_FILE_SPC1\(\)](#), [BINARY_FILE::SET_BINARY_FILE\(\)](#), [BINARY_FILE::SKIP_BINARY_FILE\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_CHARACTER\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_DP\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_DP1\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_DPC\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_DPC1\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_INTG\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_LINTG\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_LINTG1\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_LOGICAL\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_SINTG\(\)](#), [BINARY_FILE::WRITE_BINARY_FILE_SINTG1\(\)](#),

`BINARY_FILE::WRITE_BINARY_FILE_SP()`, `BINARY_FILE::WRITE_BINARY_FILE_SP1()`,
`BINARY_FILE::WRITE_BINARY_FILE_SPC()`, and `BINARY_FILE::WRITE_BINARY_FILE_-SPC1()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.17.2.2 INTEGER(INTG) F90C::CSTRINGLENGTH (INTEGER(INTG),dimension(*),intent(in) CSTRING)

Definition at line 85 of file f90c_f.f90.

Referenced by `C2FSTRING()`, and `BINARY_FILE::READ_BINARY_FILE_CHARACTER()`.

Here is the caller graph for this function:

6.17.2.3 subroutine F90C::F2CString (INTEGER(INTG),dimension(:,intent(out) CSTRING, CHARACTER(LEN=*),intent(in) FSTRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *, INTEGER(INTG),intent(in),optional FSTRINGLEN)

Definition at line 174 of file f90c_f.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, `FSTRINGLENGTH()`, and `MACHINE_CONSTANTS::INTEGER_SIZE`.

Referenced by `BINARY_FILE::OPEN_BINARY_FILE()`, and `BINARY_FILE::WRITE_BINARY_-FILE_CHARACTER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.17.2.4 INTEGER(INTG) F90C::FSTRINGLENGTH (CHARACTER(LEN=*),intent(in) FSTRING)

Definition at line 151 of file f90c_f.f90.

Referenced by `F2CString()`.

Here is the caller graph for this function:

6.18 FIELD_IO_ROUTINES Namespace Reference

Implements lists of Field IO operation.

Classes

- struct [MESH_ELEMENTS_TYPE_PTR_TYPE](#)
field variable component type pointer for IO
- struct [FIELD_VARIABLE_COMPONENT_PTR_TYPE](#)
field variable component type pointer for IO
- struct [FIELD_IO_INFO_SET](#)
contains information for parallel IO, and it is nodal base
- struct [FIELD_IO_NODAL_INFO_SET](#)
contains information for parallel IO, and it is nodal base
- struct [FIELD_IO_ELEMENTALL_INFO_SET](#)
contains information for parallel IO, and it is nodal base

Functions

- subroutine [FIELD_IO_FIELD_INFO](#) (STRING, LABEL_TYPE, FIELD_TYPE, ERR, ERROR,*)
Get the field information.
- INTEGER(INTG) [FIELD_IO_DERIVATIVE_INFO](#) (LINE, ERR, ERROR)
Get the derivative information.
- subroutine [FIELD_IO_CREATE_FIELDS](#)
Create decompsition.
- subroutine [FIE](#) (DECOMPOSITION, DECOMPOSITION_USER_NUMBER, DECOMPOSITION_METHOD, MESH, NUMBER_OF_DOMAINS,&ERR, ERROR,*)
Create decompiton.
- subroutine [FIELD_IO_FILEDS_IMPORT](#) (NAME, METHOD, REGION, MESH, MESH_USER_NUMBER, DECOMPOSITION, DECOMPOSITION_USER_NUMBER,&DECOMPOSITION_METHOD, FIELD_VALUES_SET_TYPE, FIELD_SCALING_TYPE, ERR, ERROR,*)
Import fields from files into different computational nodes.
- subroutine [FIELD_IO_FILL_BASIS_INFO](#) (INTERPOLATION_XI, LIST_STR, NUMBER_OF_COMPONENTS, ERR, ERROR,*)
Finding basis information.

- subroutine `FIELD_IO_IMPORT_GLOBAL_MESH` (NAME, REGION, MESH, MESH_USER_NUMBER, MESH_TER_COMPUTATIONAL_NUMBER,&my_computational_node_number,&MESH_COMPONENTS_OF_FIELD_COMPONENTS,&COMPONENTS_IN_FIELDS, NUMBER_OF_FIELDS, NUMBER_OF_EXNODE_FILES, ERR, ERROR,*)

Read the global mesh into one computational node first and then broadcasting to others nodes.

- subroutine `FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE` (INTERPOLATION, LABEL_TYPE, ERR, ERROR,*)

Finding basis information.

- subroutine `FIELD_IO_ELEMENTS_EXPORT` (FIELDS, FILE_NAME, METHOD, ERR, ERROR,*)

Export elemental information into multiple files.

- TYPE(`VARYING_STRING`) `FIELD_IO_BASIS_LHTP_FAMILY_LABEL` (BASIS, num_scl, num_node, LABEL_TYPE, ERR, ERROR)

Finding basis information.

- subroutine `FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, LOCAL ELEMENTAL_NUMBER, MAX_NODE_CPMP_INDEX,&NUM_OF_SCALING_FACTOR_SETS, LIST_COMP_SCALE, my_computational_node_number, FILE_ID, ERR, ERROR,*)

Write the header of a group elements using FORTRAN.

- subroutine `FIELD_IO_EXPORT_ELEMENTS_INTO_` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, NAME, my_computational_node_number,&computational_node_numbers, ERR, ERROR,*)

Write all the elemental information from LOCAL_PROCESS_NODAL_INFO_SET to exelem files.

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_SORT` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, my_computational_node_number, ERR, ERROR,*)

Sort the Elemental_info_set according to the type of field variable components.

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, ERR, ERROR,*)

Collect the elemental information from each MPI process.

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, ERR, ERROR,*)

Finalized the elemental information set.

- subroutine `FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE` (LOCAL_PROCESS_ELEMENTAL_INFO_SET, FIELDS, ERR, ERROR,*)

Initialize the elemental information set.

- subroutine `FIELD_IO_NODES_EXPORT` (FIELDS, FILE_NAME, METHOD, ERR, ERROR,*)

Export nodal information.

- subroutine `FIELD_IO_NODAL_INFO_SET_INITIALISE` (LOCAL_PROCESS_NODAL_INFO_SET, FIELDS, ERR, ERROR,*)

Initialize nodal information set.

- subroutine `FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS` (LOCAL_PROCESS_NODAL_INFO_SET, ERR, ERROR,*)

Collect nodal information from each MPI process.
- subroutine `FIELD_IO_NODAL_INFO_SET_SORT` (LOCAL_PROCESS_NODAL_INFO_SET, my_computational_node_number, ERR, ERROR,*)

Sort nodal information according to the type of field variable component.
- subroutine `FIELD_IO_NODAL_INFO_SET_FINALIZE` (LOCAL_PROCESS_NODAL_INFO_SET, ERR, ERROR,*)

Finalize nodal information set.
- TYPE(VARYING_STRING) `FIELD_IO_LABEL_DERIVATIVE_INFO_GET` (GROUP_DERIVATIVES, NUMBER_DERIVATIVES, LABEL_TYPE, ERR, ERROR)

Get the derivative information.
- TYPE(VARYING_STRING) `FIELD_IO_LABEL_FIELD_INFO_GET` (COMPONENT, LABEL_TYPE, ERR, ERROR)

Get the field information.
- subroutine `FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRA` (LOCAL_PROCESS_NODAL_INFO_SET, LOCAL_NODAL_NUMBER, MAX_NUM_OF_NODAL_DERIVATIVES,&my_computational_node_number, FILE_ID, ERR, ERROR,*)

Write the header of a group nodes using FORTRAN.
- subroutine `FIELD_IO_EXPORT_NODES_INTO_LOC` (LOCAL_PROCESS_NODAL_INFO_SET, NAME, my_computational_node_number,&computational_node_numbers, ERR, ERROR,*)

Write all the nodal information from LOCAL_PROCESS_NODAL_INFO_SET to local exnode files.
- TYPE(VARYING_STRING) `FIELD_IO_FILEDS_GROUP_INFO_GET` (FIELDS, ERR, ERROR)

Get the region label.
- TYPE(VARYING_STRING) `FIELD_IO_MULTI_FILES_INFO_GET` (computational_node_numbers, ERR, ERROR)

Get the number of files.
- subroutine `FIELD_IO_FORTRAN_FILE_READ_STRING` (FILE_ID, STRING_DATA, FILE-END, ERR, ERROR,*)

Read a string using FORTRAN IO.
- subroutine `FIELD_IO_FORTRAN_FILE_WRITE_STRING` (FILE_ID, STRING_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a string using FORTRAN IO.
- subroutine `FIELD_IO_FORTRAN_FILE_READ_DP` (FILE_ID, REAL_DATA, LEN_OF_DATA, FILE-END, ERR, ERROR,*)

Read a real data using FORTRAN IO.

- subroutine `FIELD_IO_FORTRAN_FILE_WRITE_DP` (FILE_ID, REAL_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a real data using FORTRAN IO.
- subroutine `FIELD_IO_FORTRAN_FILE_READ_INTG` (FILE_ID, INTG_DATA, LEN_OF_DATA, ERR, ERROR,*)

Read a integer data.
- subroutine `FIELD_IO_FORTRAN_FILE_WRITE_INTG` (FILE_ID, INTG_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a integer data.
- subroutine `FIELD_IO_FORTRAN_FILE_OPEN` (FILE_ID, FILE_NAME, FILE_STATUS, ERR, ERROR,*)

Open a file using Fortran.
- subroutine `FIELD_IO_FORTRAN_FILE_REWIND` (FILE_ID, ERR, ERROR,*)

Close a file using FORTRAN IO to rewind.
- subroutine `FIELD_IO_FORTRAN_FILE_CLOSE` (FILE_ID, ERR, ERROR,*)

Close a file using Fortran.
- subroutine `STRING_TO_MUTI_INTEGERS_VS` (STRING, NUMBER_OF_INTEGERS, INTG_DATA, ERR, ERROR,*)
- subroutine `STRING_TO_MUTI_REALS_VS` (STRING, NUMBER_OF_REALS, REAL_DATA, POSITION, ERR, ERROR,*)

Variables

- INTEGER(INTG), parameter `SHAPE_SIZE` = 3

size of shape
- INTEGER(INTG), parameter `FIELD_IO_FIELD_LABEL` = 1

Type for label.
- INTEGER(INTG), parameter `FIELD_IO_VARIABLE_LABEL` = 2
- INTEGER(INTG), parameter `FIELD_IO_COMPONENT_LABEL` = 3
- INTEGER(INTG), parameter `FIELD_IO_DERIVATIVE_LABEL` = 4
- INTEGER(INTG), parameter `FIELD_IO_SCALE_FACTORS_NUMBER_TYPE` = 5

Type of scale factor.
- INTEGER(INTG), parameter `FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE` = 6

6.18.1 Detailed Description

Implements lists of Field IO operation.

6.18.2 Function Documentation

6.18.2.1 subroutine FIELD_IO_ROUTINES::FIE (TYPE(DECOMPOSITION_TYPE),pointer *DECOMPOSITION*, INTEGER(INTG),intent(in) *DECOMPOSITION_USER_NUMBER*, INTEGER(INTG),intent(in) *DECOMPOSITION_METHOD*, TYPE(MESH_TYPE),pointer *MESH*, INTEGER(INTG),intent(in) *NUMBER_OF_DOMAINS*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *) [private]

Create decomposition.

Parameters:

DECOMPOSITION decomposition type

DECOMPOSITION_USER_NUMBER user number for decomposition

DECOMPOSITION_METHOD decomposition type

MESH mesh type

NUMBER_OF_DOMAINS number of domains

Definition at line 690 of file field_IO_routines.f90.

References MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH(), MESH_ROUTINES::DECOMPOSITION_CREATE_START(), MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.2 TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_LABEL (TYPE(BASIS_TYPE),intent(in) *BASIS*, INTEGER(INTG),intent(inout) *num_scl*, INTEGER(INTG),intent(inout) *num_node*, INTEGER(INTG),intent(in) *LABEL_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Finding basis information.

Parameters:

BASIS The error string

ERR The error code

ERROR The error string

Definition at line 1853 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.3 subroutine FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS() [private]

Create decomposition.

Definition at line 252 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE()`, `CMISS_MPI::MPI_ERROR_CHECK()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.18.2.4 INTEGER(INTG) FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_INFO
 (TYPE(VARYING_STRING),intent(in) *LINE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Get the derivative information.

Parameters:

LINE Text info
ERR The error code
ERROR The error string

Definition at line 187 of file `field_IO_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.18.2.5 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
 LOCAL_PROCESS_ELEMENTAL_INFO_SET, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Collect the elemental information from each MPI process.

Parameters:

LOCAL_PROCESS_ELEMENTAL_INFO_SET nodal information in this process
ERR The error code
ERROR The error string

Definition at line 2765 of file `field_IO_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.18.2.6 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
 LOCAL_PROCESS_ELEMENTAL_INFO_SET, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Finalized the elemental information set.

Parameters:

LOCAL_PROCESS_ELEMENTAL_INFO_SET nodal information in this process

ERR The error code

ERROR The error string

Definition at line 2951 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

```
6.18.2.7 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_-
    INITIALISE (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
    LOCAL_PROCESS_ELEMENTAL_INFO_SET, TYPE(FIELDS_TYPE),pointer
    FIELDS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
    ERROR, *) [private]
```

Initialize the elemental information set.

Parameters:

`LOCAL_PROCESS_ELEMENTAL_INFO_SET` elemental information in this process

`FIELDS` the field object

ERR The error code

ERROR The error string

Definition at line 2996 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

```
6.18.2.8 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_-
    SET_SORT (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
    LOCAL_PROCESS_ELEMENTAL_INFO_SET, INTEGER(INTG),intent(in)
    my_computational_node_number, INTEGER(INTG),intent(out) ERR,
    TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Sort the Elemental_info_set according to the type of field variable components.

Parameters:

`LOCAL_PROCESS_ELEMENTAL_INFO_SET` elemental information in this process

`my_computational_node_number` local process number

ERR The error code

ERROR The error string

Definition at line 2534 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

6.18.2.9 subroutine FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT
 (TYPE(FIELDS_TYPE),pointer *FIELDS*, TYPE(VARYING_STRING),intent(inout)
FILE_NAME, TYPE(VARYING_STRING),intent(in) *METHOD*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Export elemental information into multiple files.

See also:

{FIELD_IO::FIELD_IO_ELEMENTS_EXPORT}.

Parameters:

FIELDS the field object
FILE_NAME file name
METHOD method used for IO
ERR The error code
ERROR The error string

Definition at line 1802 of file field_IO_routines.f90.

References COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.18.2.10 subroutine FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_-
 HEADER_FORTTRAN**(TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
LOCAL_PROCESS_ELEMENTAL_INFO_SET, LOCAL ELEMENTAL_NUMBER,
 INTEGER(INTG),intent(inout) *MAX_NODE_CPMP_INDEX*, &,intent(inout)
NUM_OF_SCALING_FACTOR_SETS, INTEGER(INTG),dimension(:),intent(inout)
LIST_COMP_SCALE, INTEGER(INTG),intent(in) *my_computational_node_number*,
 INTEGER(INTG),intent(in) *FILE_ID*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Write the header of a group elements using FORTTRAN.

Parameters:

LOCAL_PROCESS_ELEMENTAL_INFO_SET LOCAL_PROCESS_NODAL_INFO_SET
MAX_NODE_CPMP_INDEX MAX_NODE_INDEX
my_computational_node_number local process number
FILE_ID FILE ID
ERR The error code
ERROR The error string

Definition at line 1970 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

```
6.18.2.11 subroutine FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_-
           (TYPE(FIELD_IO_ELEMENTALL_INFO_SET),intent(inout)
            LOCAL_PROCESS_ELEMENTAL_INFO_SET, TYPE(VARYING_STRING),intent(in)
            NAME, INTEGER(INTG),intent(in) my_computational_node_number,
            &,intent(in) computational_node_numbers, INTEGER(INTG),intent(out) ERR,
            TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Write all the elemental information from LOCAL_PROCESS_NODAL_INFO_SET to exelem files.

Parameters:

LOCAL_PROCESS_ELEMENTAL_INFO_SET nodal information in this process
NAME the prefix name of file.
my_computational_node_number local process number
ERR The error code
ERROR The error string

Definition at line 2277 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

```
6.18.2.12 subroutine FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_-
           HEADER_FORTRA (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout)
            LOCAL_PROCESS_NODAL_INFO_SET, INTEGER(INTG),intent(in)
            LOCAL_NODAL_NUMBER, INTEGER(INTG),intent(inout) MAX_NUM_-
            OF_NODAL_DERIVATIVES, &,intent(in) my_computational_node_number,
            INTEGER(INTG),intent(in) FILE_ID, INTEGER(INTG),intent(out) ERR,
            TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Write the header of a group nodes using FORTRAN.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET LOCAL_PROCESS_NODAL_INFO_SET
LOCAL_NODAL_NUMBER LOCAL_NUMBER IN THE NODAL IO LIST
MAX_NUM_OF_NODAL_DERIVATIVES MAX_NUM_OF_NODAL_DERIVATIVES
FILE_ID FILE ID
ERR The error code
ERROR The error string

Definition at line 4310 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.18.2.13 subroutine FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_-
INTO_LOC (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout)
LOCAL_PROCESS_NODAL_INFO_SET, TYPE(VARYING_STRING),intent(in)
NAME, INTEGER(INTG),intent(in) my_computational_node_number,
&,intent(in) computational_node_numbers, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Write all the nodal information from LOCAL_PROCESS_NODAL_INFO_SET to local exnode files.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET nodal information in this process
NAME the prefix name of file.
my_computational_node_number local process number
ERR The error code
ERROR The error string

Definition at line 5082 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_VALUES_SET_TYPE, and KINDS::PTR.

Here is the call graph for this function:

**6.18.2.14 subroutine FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO (TYPE(VARYING_-
STRING),intent(in) STRING, INTEGER(INTG),intent(in) LABEL_TYPE,
INTEGER(INTG),intent(inout) FIELD_TYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Get the field information.

Parameters:

LABEL_TYPE identitor for information
ERR The error code
ERROR The error string

Definition at line 138 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_FIBRE_TYPE, and FIELD_ROUTINES::FIELD_-GEOMETRIC_TYPE.

Here is the call graph for this function:

**6.18.2.15 TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_-
INFO_GET (TYPE(FIELDS_TYPE),intent(in) FIELDS, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Get the region label.

Parameters:

FIELDS FILEDS

ERR The error code

ERROR The error string

Definition at line 5247 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.18.2.16 subroutine FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT

```
(TYPE(VARYING_STRING),intent(in) NAME, TYPE(VARYING_STRING),intent(in)
METHOD, TYPE(REGION_TYPE),pointer REGION, TYPE(MESH_TYPE),pointer
MESH, INTEGER(INTG),intent(in) MESH_USER_NUMBER,
TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,
INTEGER(INTG),intent(in) DECOMPOSITION_USER_NUMBER,
&,intent(in) DECOMPOSITION_METHOD, INTEGER(INTG),intent(in)
FIELD_VALUES_SET_TYPE, INTEGER(INTG),intent(in) FIELD_SCALING_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*)
```

Import fields from files into different computational nodes.

Parameters:

NAME name of input

METHOD method used for import

REGION region

MESH mesh type

MESH_USER_NUMBER user number for mesh

DECOMPOSITION decompistion

DECOMPOSITION_USER_NUMBER user number for decompistion

ERR The error code

ERROR The error string

Definition at line 727 of file field_IO_routines.f90.

References `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET()`, `COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`.

Here is the call graph for this function:

6.18.2.17 subroutine FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO

```
(INTEGER(INTG),dimension(:, :, intent(inout) INTERPOLATION_XI,
TYPE(VARYING_STRING),dimension(:, intent(inout) LIST_STR,
INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]
```

Finding basis information.

Parameters:

INTERPOLATION_XI xi interpolation type

LIST_STR label type

ERR The error code

ERROR The error string

Definition at line 798 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.18.2.18 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE
(INTEGER(INTG),intent(inout) *FILE_ID*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Close a file using Fortran.

Parameters:

FILE_ID file ID

ERR The error code

ERROR The error string

Definition at line 5683 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

**6.18.2.19 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN
(INTEGER(INTG),intent(inout) *FILE_ID*, TYPE(VARYING_STRING),intent(inout)
FILE_NAME, TYPE(VARYING_STRING),intent(in) *FILE_STATUS*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Open a file using Fortran.

Parameters:

FILE_ID file ID

FILE_NAME the name of file.

FILE_STATUS status for opening a file

ERR The error code

ERROR The error string

Definition at line 5600 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.2.20 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP
 (INTEGER(INTG),intent(in) *FILE_ID*, REAL(DP),dimension(:),intent(out)
REAL_DATA, INTEGER(INTG),intent(in) *LEN_OF_DATA*,
 LOGICAL,intent(inout) *FILE_END*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Read a real data using FORTRAN IO.

Parameters:

FILE_ID file ID
REAL_DATA the name of file.
LEN_OF_DATA length of string
ERR The error code
ERROR The error string

Definition at line 5443 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.21 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG
 (INTEGER(INTG),intent(in) *FILE_ID*, INTEGER(INTG),dimension(:),intent(out)
INTG_DATA, INTEGER(INTG),intent(in) *LEN_OF_DATA*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Read a integer data.

Parameters:

FILE_ID file ID
INTG_DATA the name of file.
LEN_OF_DATA length of string
ERR The error code
ERROR The error string

Definition at line 5544 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.22 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING
 (INTEGER(INTG),intent(in) *FILE_ID*, TYPE(VARYING_STRING),intent(inout)
STRING_DATA, LOGICAL,intent(inout) *FILE_END*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Read a string using FORTRAN IO.

Parameters:

FILE_ID file ID
STRING_DATA the string data.
FILE_END file end
ERR The error code
ERROR The error string

Definition at line 5332 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.23 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND
(INTEGER(INTG),intent(inout) FILE_ID, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Close a file using FORTRAN IO to rewind.

Parameters:

FILE_ID file ID
ERR The error code
ERROR The error string

Definition at line 5659 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.24 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP
(INTEGER(INTG),intent(in) FILE_ID, REAL(DP),dimension(:),intent(in) REAL_-
DATA, INTEGER(INTG),intent(in) LEN_OF_DATA, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Write a real data using FORTRAN IO.

Parameters:

FILE_ID file ID
REAL_DATA the name of file.
LEN_OF_DATA length of string
ERR The error code
ERROR The error string

Definition at line 5479 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.18.2.25 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG
 (INTEGER(INTG),intent(in) *FILE_ID*, INTEGER(INTG),dimension(:),intent(in)
INTG_DATA, INTEGER(INTG),intent(in) *LEN_OF_DATA*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Write a integer data.

Parameters:

FILE_ID file ID

INTG_DATA the name of file.

LEN_OF_DATA length of string

ERR The error code

ERROR The error string

Definition at line 5572 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_-ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.2.26 subroutine FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING
 (INTEGER(INTG),intent(in) *FILE_ID*, TYPE(VARYING_STRING),intent(in)
STRING_DATA, INTEGER(INTG),intent(in) *LEN_OF_DATA*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Write a string using FORTRAN IO.

Parameters:

FILE_ID file ID

STRING_DATA the string data.

LEN_OF_DATA length of string

ERR The error code

ERROR The error string

Definition at line 5375 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_-ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.2.27 subroutine FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH

```
(TYPE(VARYING_STRING),intent(in) NAME, TYPE(REGION_TYPE),pointer
REGION, TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(in)
MESH_USER_NUMBER, INTEGER(INTG),intent(inout) NUMBER_OF_FIELDS,
INTEGER(INTG),intent(inout) NUMBER_OF_EXNODE_FILES,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]
```

Read the global mesh into one computational node first and then broadcasting to others nodes.

Parameters:

NAME the name of elment file
REGION region
MESH mesh type
MESH_USER_NUMBER user number of mesh
ERR The error code
ERROR The error string

Definition at line 842 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MESH_ROUTINES::MESH_CREATE_FINISH(), MESH_ROUTINES::MESH_-CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_-FINISH(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET(), MESH_-ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(), MESH_-ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET(), CMISS_MPI::MPI_ERROR_-CHECK(), NODE_ROUTINES::NODE_NUMBER_SET(), NODE_ROUTINES::NODES_CREATE_-FINISH(), NODE_ROUTINES::NODES_CREATE_START(), and KINDS::PTR.

Here is the call graph for this function:

6.18.2.28 TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_LABEL_-DERIVATIVE_INFO_GET

```
(INTEGER(INTG),dimension(number_-derivatives),intent(in) GROUP_DERIVATIVES, INTEGER(INTG),intent(in)
NUMBER_DERIVATIVES, INTEGER(INTG),intent(in) LABEL_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]
```

Get the derivative information.

Parameters:

LABEL_TYPE identitor for information
ERR The error code
ERROR The error string

Definition at line 3761 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.18.2.29 `TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET (TYPE(FIELD_VARIABLE_COMPONENT_TYPE),pointer COMPONENT, INTEGER(INTG),intent(in) LABEL_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`
`[private]`

Get the field information.

Parameters:

`LABEL_TYPE` identitor for information
`ERR` The error code
`ERROR` The error string

Definition at line 3845 of file `field_IO_routines.f90`.

References `COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_FIBRE_TYPE`, `FIELD_ROUTINES::FIELD_GENERAL_TYPE`, `FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE`, `FIELD_ROUTINES::FIELD_MATERIAL_TYPE`, `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE`, `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`, `FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE`, and `FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE`.

Here is the call graph for this function:

6.18.2.30 `TYPE(VARYING_STRING) FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET (INTEGER(INTG),intent(in) computational_node_numbers, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`
`[private]`

Get the number of files.

Parameters:

`ERR` The error code
`ERROR` The error string

Definition at line 5274 of file `field_IO_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.18.2.31 `subroutine FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout) LOCAL_PROCESS_NODAL_INFO_SET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Collect nodal information from each MPI process.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET nodal information in this process
ERR The error code
ERROR The error string

Definition at line 3272 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.18.2.32 subroutine FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout) *LOCAL_PROCESS_NODAL_INFO_SET*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalize nodal information set.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET nodal information in this process
ERR The error code
ERROR The error string

Definition at line 3716 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.18.2.33 subroutine FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE (TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout) *LOCAL_PROCESS_NODAL_INFO_SET*, TYPE(FIELDS_TYPE),pointer *FIELDS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialize nodal information set.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET nodal information in this process
FIELDS the field object
ERR The error code
ERROR The error string

Definition at line 3186 of file field_IO_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.18.2.34 subroutine FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT
`(TYPE(FIELD_IO_NODAL_INFO_SET),intent(inout) LOCAL_PROCESS_-`
`NODAL_INFO_SET, INTEGER(INTG),intent(in) my_computational_node_number,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Sort nodal information according to the type of field variable component.

Parameters:

LOCAL_PROCESS_NODAL_INFO_SET nodal information in this process
my_computational_node_number local process number
ERR The error code
ERROR The error string

Definition at line 3463 of file field_IO_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-`
`ROUTINES::EXITS()`, and `KINDS::PTR`.

Here is the call graph for this function:

6.18.2.35 subroutine FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT
`(TYPE(FIELDS_TYPE),pointer FIELDS, TYPE(VARYING_STRING),intent(inout)`
`FILE_NAME, TYPE(VARYING_STRING),intent(in) METHOD,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Export nodal information.

See also:

{FIELD_IO::FIELD_IO_NODES_EXPORT}.

Parameters:

FIELDS the field object
FILE_NAME file name
ERR The error code
ERROR The error string

Definition at line 3135 of file field_IO_routines.f90.

References `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET()`, `COMP_-`
`ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`, `BASE_ROUTINES::ENTERS()`,
`BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.18.2.36 subroutine FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_-
`INTERPOLATION_TYPE (INTEGER(INTG),intent(inout) INTERPOLATION,`
`TYPE(VARYING_STRING),intent(in) LABEL_TYPE, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finding basis information.

Parameters:

INTERPOLATION xi interpolation type

LABEL_TYPE label type

ERR The error code

ERROR The error string

Definition at line 1620 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.2.37 subroutine FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS
(TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(in)
NUMBER_OF_INTEGERS, INTEGER(INTG),dimension(:),intent(inout) INTG_DATA,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 \ast) [private]

Definition at line 5703 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.2.38 subroutine FIELD_IO_ROUTINES::STRING_TO_MUTI_REALS_VS
(TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(in)
NUMBER_OF_REALS, REAL(DP),dimension(:),intent(inout) REAL_DATA,
INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, \ast) [private]

Definition at line 5746 of file field_IO_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.18.3 Variable Documentation

**6.18.3.1 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_COMPONENT_-
LABEL = 3**

Definition at line 78 of file field_IO_routines.f90.

**6.18.3.2 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_-
LABEL = 4**

Definition at line 79 of file field_IO_routines.f90.

6.18.3.3 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_FIELD_LABEL = 1

Type for lable.

Definition at line 76 of file field_IO_routines.f90.

**6.18.3.4 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_SCALE_-
FACTORS_NUMBER_TYPE = 5**

Type of scale factor.

Definition at line 82 of file field_IO_routines.f90.

**6.18.3.5 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_SCALE_-
FACTORS_PROPERTY_TYPE = 6**

Definition at line 83 of file field_IO_routines.f90.

**6.18.3.6 INTEGER(INTG),parameter FIELD_IO_ROUTINES::FIELD_IO_VARIABLE_LABEL
= 2**

Definition at line 77 of file field_IO_routines.f90.

6.18.3.7 INTEGER(INTG),parameter FIELD_IO_ROUTINES::SHAPE_SIZE = 3

size of shape

Definition at line 73 of file field_IO_routines.f90.

6.19 FIELD_ROUTINES Namespace Reference

This module handles all field related routines.

Classes

- interface [FIELD_COMPONENT_INTERPOLATION_SET](#)
- interface [FIELD_COMPONENT_MESH_COMPONENT_SET](#)
- interface [FIELD_DEPENDENT_TYPE_SET](#)
- interface [FIELD_DIMENSION_SET](#)
- interface [FIELD_GEOMETRIC_FIELD_SET](#)
- interface [FIELD_MESH_DECOMPOSITION_SET](#)
- interface [FIELD_NUMBER_OF_COMPONENTS_SET](#)
- interface [FIELD_NUMBER_OF_VARIABLES_SET](#)
- interface [FIELD_SCALING_TYPE_SET](#)
- interface [FIELD_TYPE_SET](#)

Functions

- subroutine [FIELD_COMPONENT_INTER](#) (USER_NUMBER, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, REGION,&INTERPOLATION_TYPE, ERR, ERROR,*)

Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.
- subroutine [FI](#) (FIELD, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, INTERPOLATION_TYPE,&ERR, ERROR,*)

Sets/changes the interpolation type for a field variable component identified by a pointer.
- subroutine [FIELD_COMPONENT_MESH_COM](#) (USER_NUMBER, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, REGION,&MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.
- subroutine [FIELD_VARIABLE_COMPONENT_FINALISE](#) (FIELD_VARIABLE_COMPONENT, ERR, ERROR,*)

Finalises a field variable component and deallocates all memory.
- subroutine [FIELD_VARIABLE_COMPONENT_INITIALISE](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, ERR, ERROR,*)

Initialises a field variable component.
- subroutine [FIELD_CREATE_FINISH](#) (REGION, FIELD, ERR, ERROR,*)

Finishes the creation of a field on a region.
- subroutine [FIELD_CREATE_VALUES_CACHE_FINALISE](#) (FIELD, ERR, ERROR,*)

Finalise the create values cache for a field.
- subroutine [FIELD_CREATE_VALUES_CACHE_INITIALISE](#) (FIELD, ERR, ERROR,*)

Initialises the create values cache for a field.

- subroutine **FIELD_DEPENDENT_TYPE_SET_NUMBER** (USER_NUMBER, REGION, DEPENDENT_TYPE, ERR, ERROR,*)

Sets/changes the dependent type for a field identified by a user number.

- subroutine **FIELD_DEPENDENT_TYPE_SET_PTR** (FIELD, DEPENDENT_TYPE, ERR, ERROR,*)

Sets/changes the dependent type for a field identified by a pointer.

- subroutine **FIELD_DESTROY** (FIELD, ERR, ERROR,*)

Destroys a field identified by a pointer to a field.

- subroutine **FIELD_DIMENSION_SET_NUMBER** (USER_NUMBER, REGION, FIELD_DIMENSION, ERR, ERROR,*)

Sets/changes the field dimension for a field identified by a user number.

- subroutine **FIELD_DIMENSION_SET_PTR** (FIELD, FIELD_DIMENSION, ERR, ERROR,*)

Sets/changes the field dimension for a field identified by a pointer.

- subroutine **FIELD_INTERPOLATE_GAUSS** (PARTIAL_DERIVATIVE_TYPE, QUADRATURE_SCHEME, GAUSS_POINT_NUMBER, INTERPOLATED_POINT, ERR, ERROR,*)

Interpolates a field at a gauss point to give an interpolated point. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.

- subroutine **FIELD_INTERPOLATE_XI** (PARTIAL_DERIVATIVE_TYPE, XI, INTERPOLATED_POINT, ERR, ERROR,*)

Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.

- subroutine **FIELD_INTERPOLATED_POINT_FINALISE** (INTERPOLATED_POINT, ERR, ERROR,*)

Finalises the interpolated point and deallocates all memory.

- subroutine **FIELD_INTERPOLATED_POINT_INITIALISE** (INTERPOLATION_PARAMETERS, INTERPOLATED_POINT, ERR, ERROR,*)

Initialises the interpolated point for an interpolation parameters.

- subroutine **FIELD_INTERPOLATED_POINT_METRICS_CALCULATE** (JACOBIAN_TYPE, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Calculates the interpolated point metrics and the associated interpolated point.

- subroutine **FIELD_INTERPOLATED_POINT_METRICS_FINALISE** (INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Finalises the interpolated point metrics and deallocates all memory.

- subroutine **FIELD_INTERPOLATED_POINT_METRICS_INITIALISE** (INTERPOLATED_POINT, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Initialises the interpolated point metrics for an interpolated point.
- subroutine **FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET** (PARAMETER_SET_NUMBER, ELEMENT_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Gets the interpolation parameters for a particular element. Old CMISS name XPXE, ZPZE.
- subroutine **FIELD_INTERPOLATION_PARAMETERS_FINALISE** (INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Finalises the interpolation parameters and deallocates all memory.
- subroutine **FIELD_INTERPOLATION_PARAMETERS_INITIALISE** (FIELD, VARIABLE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Initialises the interpolation parameters for a field variable.
- subroutine **FIELD_INTERPOLATION_PARAMETERS_LINE_GET** (PARAMETER_SET_NUMBER, LINE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Gets the interpolation parameters for a particular line. Old CMISS name XPXE, ZPZE.
- subroutine **FIELD_VARIABLES_FINALISE** (FIELD, ERR, ERROR,*)

Finalises the field variables for a field and deallocates all memory.
- subroutine **FIELD_VARIABLES_INITIALISE** (FIELD, ERR, ERROR,*)

Initialises the field variables.
- subroutine **FIELDS_FINALISE** (REGION, ERR, ERROR,*)

Finalises the fields for the given region and deallocates all memory.
- subroutine **FIELDS_INITIALISE** (REGION, ERR, ERROR,*)

Initialises the fields for the given region.

Variables

- INTEGER(INTG), parameter **FIELD_INDEPENDENT_TYPE** = 1

Independent field type.
- INTEGER(INTG), parameter **FIELD_DEPENDENT_TYPE** = 2

Dependent field type.
- INTEGER(INTG), parameter **FIELD_SCALAR_DIMENSION_TYPE** = 1

Scalar field.
- INTEGER(INTG), parameter **FIELD_VECTOR_DIMENSION_TYPE** = 2

Vector field.
- INTEGER(INTG), parameter **FIELD_GEOMETRIC_TYPE** = 1

Geometric field.

- INTEGER(INTG), parameter **FIELD_FIBRE_TYPE** = 2
Fibre field.
- INTEGER(INTG), parameter **FIELD_GENERAL_TYPE** = 3
General field.
- INTEGER(INTG), parameter **FIELD_MATERIAL_TYPE** = 4
Material field.
- INTEGER(INTG), parameter **FIELD_CONSTANT_INTERPOLATION** = 1
Constant interpolation. One parameter for the field.
- INTEGER(INTG), parameter **FIELD_ELEMENT_BASED_INTERPOLATION** = 2
Element based interpolation. Parameters are different in each element.
- INTEGER(INTG), parameter **FIELD_NODE_BASED_INTERPOLATION** = 3
Node based interpolation. Parameters are nodal based and a basis function is used.
- INTEGER(INTG), parameter **FIELD_POINT_BASED_INTERPOLATION** = 4
Point based interpolation. Parameters are different at each grid point.
- INTEGER(INTG), parameter **FIELD_NUMBER_OF_VARIABLE_TYPES** = 4
Number of different field variable types possible.
- INTEGER(INTG), parameter **FIELD_STANDARD_VARIABLE_TYPE** = 1
Standard variable type i.e., u .
- INTEGER(INTG), parameter **FIELD_NORMAL_VARIABLE_TYPE** = 2
Normal derivative variable type i.e., du/dn .
- INTEGER(INTG), parameter **FIELD_TIME_DERIV1_VARIABLE_TYPE** = 3
First time derivative variable type i.e., du/dt .
- INTEGER(INTG), parameter **FIELD_TIME_DERIV2_VARIABLE_TYPE** = 4
Second type derivative variable type i.e., d^2u/dt^2 .
- INTEGER(INTG), parameter **FIELD_CONSTANT_DOF_TYPE** = 1
The dof is from a field variable component with constant interpolation.
- INTEGER(INTG), parameter **FIELD_ELEMENT_DOF_TYPE** = 2
The dof is from a field variable component with element based interpolation.
- INTEGER(INTG), parameter **FIELD_NODE_DOF_TYPE** = 3
The dof is from a field variable component with node based interpolation.
- INTEGER(INTG), parameter **FIELD_POINT_DOF_TYPE** = 4
The dof is from a field variable component with point based interpolation.
- INTEGER(INTG), parameter **FIELD_NUMBER_OF_SET_TYPES** = 99

The maximum number of different parameter sets for a field.

- INTEGER(INTG), parameter **FIELD_VALUES_SET_TYPE** = 1
The parameter set corresponding to the field values.
- INTEGER(INTG), parameter **FIELD_BOUNDARY_CONDITIONS_SET_TYPE** = 2
The parameter set corresponding to the field boundary conditions.
- INTEGER(INTG), parameter **FIELD_INITIAL_CONDITIONS_SET_TYPE** = 3
The parameter set corresponding to the field initial conditions.
- INTEGER(INTG), parameter **FIELD_NO_SCALING** = 0
The field is not scaled.
- INTEGER(INTG), parameter **FIELD_UNIT_SCALING** = 1
The field has unit scaling.
- INTEGER(INTG), parameter **FIELD_ARC_LENGTH_SCALING** = 2
The field has arc length scaling.
- INTEGER(INTG), parameter **FIELD_ARITHMETIC_MEAN_SCALING** = 3
The field has arithmetic mean of the arc length scaling.
- INTEGER(INTG), parameter **FIELD_HARMONIC_MEAN_SCALING** = 4
The field has geometric mean of the arc length scaling.

6.19.1 Detailed Description

This module handles all field related routines.

6.19.2 Function Documentation

6.19.2.1 subroutine FIELD_ROUTINES::FI (TYPE(FIELD_TYPE),pointer **FIELD**, INTEGER(INTG),intent(in) **FIELD_VARIABLE_NUMBER**, INTEGER(INTG),intent(in) **FIELD_COMPONENT_NUMBER**, INTEGER(INTG),intent(in) **INTERPOLATION_TYPE**, &,intent(out) **ERR**, INTEGER(INTG),intent(out) **error**, *) [private]

Sets/changes the interpolation type for a field variable component identified by a pointer.

Sets/changes the mesh component number for a field variable component identified by a pointer to a field and a field variable number.

Parameters:

FIELD A pointer to the field to set the interpolation for

FIELD_VARIABLE_NUMBER The field variable number of the field variable component to set

FIELD_COMPONENT_NUMBER The field component number of the field variable component to set

INTERPOLATION_TYPE The interpolation type to set

See also:

[FIELD_ROUTINES::VariableTypes](#), [FIELD_ROUTINES](#)

Definition at line 276 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_CONSTANT_INTERPOLATION`, `FIELD_ELEMENT_BASED_INTERPOLATION`, `FIELD_NODE_BASED_INTERPOLATION`, and `FIELD_POINT_BASED_INTERPOLATION`.

Here is the call graph for this function:

```
6.19.2.2 subroutine FIELD_ROUTINES::FIELD_COMPONENT_INTER
  (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(in) FIELD_VARIABLE_NUMBER, INTEGER(INTG),intent(in) FIELD_COMPONENT_NUMBER,
   TYPE(REGION_TYPE),pointer REGION, &,intent(in) INTERPOLATION_TYPE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
  [private]
```

Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.

Parameters:

FIELD_VARIABLE_NUMBER The field variable number of the field variable component
FIELD_COMPONENT_NUMBER The field component number of the field variable component
REGION The region containing the field
ERR The error code
ERROR The error string

Definition at line 241 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.19.2.3 subroutine FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM
  (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(in) FIELD_VARIABLE_NUMBER, INTEGER(INTG),intent(in) FIELD_COMPONENT_NUMBER,
   TYPE(REGION_TYPE),pointer REGION, &,intent(in) MESH_COMPONENT_NUMBER, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *)
  [private]
```

Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.

Todo

change variable number to variable type.

Parameters:

USER_NUMBER The user number of the field to set the mesh component for

FIELD_VARIABLE_NUMBER The field variable number of the field variable component to set
FIELD_COMPONENT_NUMBER The field component number of the field variable component to set
REGION A pointer to the region containing the field
ERR The error code
ERROR The error string

Definition at line 337 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_CONSTANT_INTERPOLATION`, `FIELD_ELEMENT_BASED_INTERPOLATION`, `FIELD_NODE_BASED_INTERPOLATION`, and `FIELD_POINT_BASED_INTERPOLATION`.

Here is the call graph for this function:

6.19.2.4 subroutine `FIELD_ROUTINES::FIELD_CREATE_FINISH` (`TYPE(REGION_TYPE)`, pointer *REGION*, `TYPE(FIELD_TYPE)`,pointer *FIELD*, `INTEGER(INTG)`,intent(out) *ERR*, `TYPE(VARYING_STRING)`,intent(out) *ERROR*, *)

Finishes the creation of a field on a region.

Parameters:

REGION A pointer to the region containing the field
FIELD A pointer to the field to finish the creation of
ERR The error code
ERROR The error string

Definition at line 743 of file field_routines.f90.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`,
`BASE_ROUTINES::EXITS()`, `FIELD_ARITHMETIC_MEAN_SCALING`, `FIELD_CREATE_VALUES_CACHE_FINALISE()`, `FIELD_CREATE_VALUES_CACHE_INITIALISE()`, `FIELD_GEOMETRIC_TYPE`, `FIELD_INDEPENDENT_TYPE`, `FIELD_NUMBER_OF_VARIABLE_TYPES`, `FIELD_VALUES_SET_TYPE`, `FIELD_VARIABLES_INITIALISE()`, `FIELD_VECTOR_DIMENSION_TYPE`, and `KINDS::PTR`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS()`, and `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.5 subroutine `FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE` (`TYPE(FIELD_TYPE)`,pointer *FIELD*, `INTEGER(INTG)`,intent(out) *ERR*, `TYPE(VARYING_STRING)`,intent(out) *ERROR*, *) [private]

Finalise the create values cache for a field.

Parameters:

FIELD A pointer to the field to finalise the create values cache for

ERR The error code

ERROR The error string

Definition at line 924 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_CREATE_FINISH()`, and `FIELD_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.6 subroutine `FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE`

```
(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR,
     TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Initialises the create values cache for a field.

Parameters:

FIELD A pointer to the field to initialise the create values cache for

ERR The error code

ERROR The error string

Definition at line 957 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_FIBRE_TYPE`, `FIELD_GENERAL_TYPE`, `FIELD_GEOMETRIC_TYPE`, `FIELD_MATERIAL_TYPE`, and `FIELD_NODE_BASED_INTERPOLATION`.

Referenced by `FIELD_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.7 subroutine `FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_NUMBER`

```
(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_-_
      TYPE),pointer REGION, INTEGER(INTG),intent(in) DEPENDENT_TYPE,
      INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
     [private]
```

Sets/changes the dependent type for a field identified by a user number.

Parameters:

USER_NUMBER The user number of the field to set the dependent type for

REGION A pointer to the region containing the field

DEPENDENT_TYPE The dependent type to set/change for the field

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1015 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `FIELD_DEPENDENT_TYPE_SET_PTR()`.

Here is the call graph for this function:

**6.19.2.8 subroutine FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR
(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(in) DEPENDENT_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Sets/changes the dependent type for a field indentified by a pointer.

Parameters:

FIELD A pointer to the field to set/change the dependent type for

DEPENDENT_TYPE The dependent type to set/change

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1046 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_DEPENDENT_TYPE`, and `FIELD_INDEPENDENT_TYPE`.

Referenced by `FIELD_DEPENDENT_TYPE_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.9 subroutine FIELD_ROUTINES::FIELD_DESTROY (TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Destroys a field identified by a pointer to a field.

Parameters:

FIELD A pointer to the field to destroy

ERR The error code

ERROR The error string

Definition at line 1121 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_CREATE_VALUES_CACHE_FINALISE()`, `FIELD_VARIABLES_FINALISE()`, and `KINDS::PTR`.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_FINALISE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_FINALISE(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE(), and FIELDS_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.10 subroutine FIELD_ROUTINES::FIELD_DIMENSION_SET_NUMBER
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(in) FIELD_DIMENSION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sets/changes the field dimension for a field identified by a user number.

Parameters:

USER_NUMBER The user number of the field to set the dimension for

REGION A pointer to the region containing the field

FIELD_DIMENSION The dimension to set/change

See also:

[FIELD_ROUTINES::DimensionTypes](#), [FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1219 of file field_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and FIELD_DIMENSION_SET_PTR().

Here is the call graph for this function:

6.19.2.11 subroutine FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR (TYPE(FIELD_TYPE),pointer *FIELD*, INTEGER(INTG),intent(in) *FIELD_DIMENSION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets/changes the field dimension for a field identified by a pointer.

Parameters:

FIELD A pointer to the field to set/change the dimension for

FIELD_DIMENSION The field dimension to set/change

See also:

[FIELD_ROUTINES::DimensionTypes](#), [FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1250 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_SCALAR_DIMENSION_TYPE`, and `FIELD_VECTOR_DIMENSION_TYPE`.

Referenced by `FIELD_DIMENSION_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.12 subroutine `FIELD_ROUTINES::FIELD_INTERPOLATE_GAUSS`
`(INTEGER(INTG),intent(in) PARTIAL_DERIVATIVE_TYPE,`
`INTEGER(INTG),intent(in) QUADRATURE_SCHEME, INTEGER(INTG),intent(in)`
`GAUSS_POINT_NUMBER, TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer`
`INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Interpolates a field at a gauss point to give an interpolated point. `PARTIAL_DERIVATIVE_TYPE` controls which partial derivatives are evaluated. If it is `NO_PART_DERIV` then only the field values are interpolated. If it is `FIRST_PART_DERIV` then the field values and first partial derivatives are interpolated. If it is `SECOND_PART_DERIV` the the field values and first and second partial derivatives are evaluated. Old [CMISS](#) name `XEXG`, `ZEXG`.

Parameters:

`PARTIAL_DERIVATIVE_TYPE` The partial derivative type of the provided field interpolation

`QUADRATURE_SCHEME` The quadrature scheme of the Gauss points

See also:

`BASIS_ROUTINES_QuadratureSchemes,BASIS_ROUTINES`

`GAUSS_POINT_NUMBER` The number of the Gauss point to interpolate the field at

`INTERPOLATED_POINT` The pointer to the interpolated point which will contain the field interpolation information at the specified Gauss point

`ERR` The error code

`ERROR` The error string

Definition at line 1322 of file `field_routines.f90`.

References `COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.13 subroutine FIELD_ROUTINES::FIELD_INTERPOLATE_XI
(INTEGER(INTG),intent(in) PARTIAL_DERIVATIVE_TYPE,
REAL(DP),dimension(:),intent(in) XI, TYPE(FIELD_INTERPOLATED_POINT_-
TYPE),pointer INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.

Parameters:

PARTIAL_DERIVATIVE_TYPE The partial derivative type of the provide field interpolation
XI XI(ni). The ni'th Xi coordinate to evaluate the field at
INTERPOLATED_POINT The pointer to the interpolated point which will contain the field interpolation information at the specified Xi point
ERR The error code
ERROR The error string

Definition at line 1407 of file field_routines.f90.

References COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.14 subroutine FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE
(TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer INTERPOLATED_POINT,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***)**

Finalises the interpolated point and deallocates all memory.

Parameters:

INTERPOLATED_POINT A pointer to the interpolated point to finalise
ERR The error code
ERROR The error string

Definition at line 1502 of file field_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_FINALISE(), FIELD_INTERPOLATED_POINT_INITIALISE(), and FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.15 subroutine FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE
(TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
INTERPOLATION_PARAMETERS, TYPE(FIELD_INTERPOLATED_POINT_-
TYPE),pointer INTERPOLATED_POINT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Initialises the interpolated point for an interpolation parameters.

Parameters:

INTERPOLATION_PARAMETERS A pointer to the interpolation parameters to initialise the interpolated point for

INTERPOLATED_POINT On exit, A pointer to the interpolated point that has been initialised

ERR The error code

ERROR The error string

Definition at line 1529 of file field_routines.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and FIELD_INTERPOLATED_POINT_FINALISE().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE(), and FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.16 subroutine FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_-
METRICS_CALCULATE (INTEGER(INTG),intent(in) JACOBIAN_TYPE,
TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE),pointer
INTERPOLATED_POINT_METRICS, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Calculates the interpolated point metrics and the associated interpolated point.

Parameters:

JACOBIAN_TYPE The Jacobian type of the calculation

See also:

COORDINATE_ROUTINES_JacobianTypes, [COORDINATE_ROUTINES](#)

INTERPOLATED_POINT_METRICS A pointer to the interpolated point metrics

ERR The error code

ERROR The error string

Definition at line 1577 of file field_routines.f90.

References COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_FIBRE_TYPE, and FIELD_GEOMETRIC_TYPE.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.17 subroutine FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_FINALISE (TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE),pointer *INTERPOLATED_POINT_METRICS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finalises the interpolated point metrics and deallocates all memory.

Parameters:

INTERPOLATED_POINT_METRICS A pointer to the interpolated point metrics to finalise

ERR The error code

ERROR The error string

Definition at line 1618 of file field_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_FINALISE(), and FIELD_INTERPOLATED_POINT_METRICS_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.18 subroutine FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE (TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer *INTERPOLATED_POINT*, TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE),pointer *INTERPOLATED_POINT_METRICS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Initialises the interpolated point metrics for an interpolated point.

Parameters:

INTERPOLATED_POINT_METRICS On exit, a pointer to the interpolated point metrics that have been initialised

ERR The error code

ERROR The error string

Definition at line 1648 of file field_routines.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and FIELD_INTERPOLATED_POINT_METRICS_FINALISE().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.19.2.19 subroutine FIELD_ROUTINES::FIELD_INTERPOLATION_-
PARAMETERS_ELEMENT_GET (INTEGER(INTG),intent(in)
PARAMETER_SET_NUMBER, INTEGER(INTG),intent(in) ELEMENT_NUMBER,
TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
INTERPOLATION_PARAMETERS, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Gets the interpolation parameters for a particular element. Old [CMISS](#) name XPXE, ZPZE.

Parameters:

PARAMETER_SET_NUMBER The field parameter set number to get the element parameters for
ELEMENT_NUMBER The element number to get the element parameters for
INTERPOLATION_PARAMETERS A pointer to the interpolation parameters
ERR The error code
ERROR The error string

Definition at line 1712 of file field_routines.f90.

References COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_-
 PARAMETERS_ADJUST(), BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-
 ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
 BASE_ROUTINES::EXITS(), FIELD_ARC_LENGTH_SCALING, FIELD_ARITHMETIC_MEAN_-
 SCALING, FIELD_CONSTANT_INTERPOLATION, FIELD_ELEMENT_BASED_INTERPOLATION,
 FIELD_HARMONIC_MEAN_SCALING, FIELD_NO_SCALING, FIELD_NODE_BASED_-
 INTERPOLATION, FIELD_POINT_BASED_INTERPOLATION, FIELD_UNIT_SCALING, and
 KINDS::PTR.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_-
 FINITE_ELEMENT_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.19.2.20 subroutine FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_-
FINALISE (TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
INTERPOLATION_PARAMETERS, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Finalises the interpolation parameters and deallocates all memory.

Parameters:

INTERPOLATION_PARAMETERS A pointer to the interpolation parameters to finalise
ERR The error code
ERROR The error string

Definition at line 1866 of file field_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-
 ROUTINES::EXITS().

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_FINALISE(),
 FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), and FIELD_INTERPOLATION_-
 PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.19.2.21 subroutine FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_-
INITIALISE (TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(in)
VARIABLE_NUMBER, TYPE(FIELD_INTERPOLATION_PARAMETERS_-
TYPE),pointer INTERPOLATION_PARAMETERS, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Initialises the interpolation parameters for a field variable.

Parameters:

FIELD A pointer to the field to initialise the interpolation parameters for

VARIABLE_NUMBER The field variable number to initialise the interpolation parameters for

INTERPOLATION_PARAMETERS On exit, a pointer to the initialised interpolation parameters.

ERR The error code

ERROR The error string

Definition at line 1895 of file field_routines.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), FIELD_INTERPOLATION_PARAMETERS_FINALISE(), FIELD_NUMBER_-OF_VARIABLE_TYPES, and KINDS::PTR.

Referenced by EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE(), and FIELD_INTERPOLATION_PARAMETERS_LINE_GET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.19.2.22 subroutine FIELD_ROUTINES::FIELD_INTERPOLATION_-
PARAMETERS_LINE_GET (INTEGER(INTG),intent(in)
PARAMETER_SET_NUMBER, INTEGER(INTG),intent(in) LINE_NUMBER,
TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
INTERPOLATION_PARAMETERS, INTEGER(INTG),intent(out),parameter ERR,
TYPE(VARYING_STRING),intent(out),parameter ERROR, *)**

Gets the interpolation parameters for a particular line. Old [CMISS](#) name XPXE, ZPZE.

Parameters:

PARAMETER_SET_NUMBER The field parameter set number to get the line parameters for

LINE_NUMBER The line number to get the line parameters for

INTERPOLATION_PARAMETERS A pointer to the interpolation parameters

ERR The error code

ERROR The error string

Definition at line 1967 of file field_routines.f90.

References KINDS::DP, COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET(), COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM(), COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST(), BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::DIAGNOSTICS2, DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST, DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL, DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_INITIALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ARC_LENGTH_SCALING, FIELD_ARITHMETIC_MEAN_SCALING, FIELD_CONSTANT_DOF_TYPE, FIELD_CONSTANT_INTERPOLATION, FIELD_DEPENDENT_TYPE, FIELD_ELEMENT_BASED_INTERPOLATION, FIELD_ELEMENT_DOF_TYPE, FIELD_FIBRE_TYPE, FIELD_GENERAL_TYPE, FIELD_GEOMETRIC_TYPE, FIELD_HARMONIC_MEAN_SCALING, FIELD_INDEPENDENT_TYPE, FIELD_INTERPOLATE_XI(), FIELD_INTERPOLATED_POINT_FINALISE(), FIELD_INTERPOLATED_POINT_INITIALISE(), FIELD_INTERPOLATION_PARAMETERS_FINALISE(), FIELD_INTERPOLATION_PARAMETERS_INITIALISE(), FIELD_MATERIAL_TYPE, FIELD_NO_SCALING, FIELD_NODE_BASED_INTERPOLATION, FIELD_NODE_DOF_TYPE, FIELD_NUMBER_OF_SET_TYPES, FIELD_POINT_BASED_INTERPOLATION, FIELD_SCALAR_DIMENSION_TYPE, FIELD_STANDARD_VARIABLE_TYPE, FIELD_UNIT_SCALING, FIELD_VALUES_SET_TYPE, FIELD_VECTOR_DIMENSION_TYPE, LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INTG_TYPE, LISTS::LIST_REMOVE_DUPLICATES(), MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE, and KINDS::PTR.

Here is the call graph for this function:

**6.19.2.23 subroutine FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE (TYPE(FIELD_VARIABLE_COMPONENT_TYPE)
FIELD_VARIABLE_COMPONENT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalises a field variable component and deallocates all memory.

Parameters:

FIELD_VARIABLE_COMPONENT The field variable component to finalise

ERR The error code

ERROR The error string

Definition at line 462 of file field_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by FIELD_VARIABLE_COMPONENT_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.24 subroutine FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE
`(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(in) VARIABLE_NUMBER, INTEGER(INTG),intent(in) COMPONENT_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises a field variable component.

Parameters:

FIELD A pointer to the field containing the field variable component to initialise
VARIABLE_NUMBER The field variable number of the field variable component
COMPONENT_NUMBER The field component number of the field variable component
ERR The error code
ERROR The error string

Definition at line 486 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_CONSTANT_INTERPOLATION`, `FIELD_ELEMENT_BASED_INTERPOLATION`, `FIELD_NODE_BASED_INTERPOLATION`, `FIELD_POINT_BASED_INTERPOLATION`, `FIELD_VARIABLE_COMPONENT_FINALISE()`, and `KINDS::PTR`.

Here is the call graph for this function:

6.19.2.25 subroutine FIELD_ROUTINES::FIELD_VARIABLES_FINALISE
`(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises the field variables for a field and deallocates all memory.

Parameters:

FIELD A pointer to the field to finalise the variables for
ERR The error code
ERROR The error string

Definition at line 5419 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.26 subroutine FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE
`(TYPE(FIELD_TYPE),pointer FIELD, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the field variables.

Parameters:

FIELD A pointer to the field to initialise the variables for
ERR The error code
ERROR The error string

Definition at line 5454 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.27 subroutine `FIELD_ROUTINES::FIELDS_FINALISE` (`TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`)

Finalises the fields for the given region and deallocates all memory.

Parameters:

REGION A pointer to the region to finalise the fields for
ERR The error code
ERROR The error string

Definition at line 5491 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_DESTROY()`, and `KINDS::PTR`.

Referenced by `REGION_ROUTINES::REGION_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.19.2.28 subroutine `FIELD_ROUTINES::FIELDS_INITIALISE` (`TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */`)

Initialises the fields for the given region.

Parameters:

REGION A pointer to the region to initialise the fields for
ERR The error code
ERROR The error string

Definition at line 5526 of file field_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by REGION_ROUTINES::REGION_CREATE_START(), and REGION_-ROUTINES::REGION_SUB_REGION_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.20 FINITE_ELEMENT_ROUTINES Namespace Reference

This module contains all finite element base routines.

Functions

- subroutine `FEM_ELEMENT_MATRICES_FINALISE` (`ELEMENT_MATRICES`, `ERR`, `ERROR,*`)
- subroutine `FEM_ELEMENT_MATRICES_INITIALISE` (`PROBLEM`, `ELEMENT_MATRICES`, `ERR`, `ERROR,*`)

6.20.1 Detailed Description

This module contains all finite element base routines.

6.20.2 Function Documentation

**6.20.2.1 subroutine `FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_-
MATRICES_FINALISE` (`TYPE(FEM_ELEMENT_MATRICES_TYPE),pointer
ELEMENT_MATRICES`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`)**

Definition at line 72 of file `finite_element_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.20.2.2 subroutine `FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_-
MATRICES_INITIALISE` (`TYPE(PROBLEM_TYPE),pointer PROBLEM`,
`TYPE(FEM_ELEMENT_MATRICES_TYPE),pointer ELEMENT_MATRICES`,
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)
[private]**

Definition at line 104 of file `finite_element_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Here is the call graph for this function:

6.21 FLUID_MECHANICS_ROUTINES Namespace Reference

This module handles all fluid mechanics class routines.

6.21.1 Detailed Description

This module handles all fluid mechanics class routines.

6.22 GENERATED_MESH_ROUTINES Namespace Reference

This module handles all generated mesh routines.

Functions

- subroutine [GENERATED_MESH_CREATE_FINISH](#) (GENERATED_MESH, ERR, ERROR,*)

Finishes the creation of a generated mesh.
- subroutine [GENERATED_MESH_CREATE_START](#) (USER_NUMBER, REGION, GENERATED_MESH, ERR, ERROR,*)

Starts the creation of a generated mesh.
- subroutine [GENERATED_MESH_DESTROY](#) (GENERATED_MESH, ERR, ERROR,*)

Destroys a generated mesh.
- subroutine [GENERATED_MESH_FINALISE](#) (GENERATED_MESH, ERR, ERROR,*)

Finalises a generated mesh and deallocates all memory.
- subroutine [GENERATED_MESH_INITIALISE](#) (GENERATED_MESH, ERR, ERROR,*)

Initialises a generated mesh.
- subroutine [GENERATED_MESH_TYPE_SET](#) (GENERATED_MESH, GENERATED_TYPE, ERR, ERROR,*)

Sets/changes the type of a generated mesh.

Variables

- INTEGER(INTG), parameter [GENERATED_MESH_REGULAR_MESH_TYPE](#) = 1

A regular generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_POLAR_MESH_TYPE](#) = 2

A polar generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_FRACTAL_TREE_MESH_TYPE](#) = 3

A fractal tree generated mesh.

6.22.1 Detailed Description

This module handles all generated mesh routines.

6.22.2 Function Documentation

**6.22.2.1 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-
FINISH (TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]**

Finishes the creation of a generated mesh.

Parameters:

GENERATED_MESH A pointer to the generated mesh to finish the creation of
ERR The error code
ERROR The error string

Definition at line 85 of file generated_mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.22.2.2 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_-
START (INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer
REGION, TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]**

Starts the creation of a generated mesh.

Parameters:

USER_NUMBER The user number of the generated mesh to create
REGION A pointer to the region to create the generated mesh on
GENERATED_MESH On exit, a pointer to the created generated mesh. Must not be associated on entry.
ERR The error code
ERROR The error string

Definition at line 112 of file generated_mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.22.2.3 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY
(TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]**

Destroys a generated mesh.

Parameters:

GENERATED_MESH A pointer to the generated mesh to destroy
ERR The error code
ERROR The error string

Definition at line 145 of file generated_mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.22.2.4 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE`
`(TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Finalises a generated mesh and deallocates all memory.

Parameters:

GENERATED_MESH A pointer to the generated mesh to finalise.
ERR The error code
ERROR The error string

Definition at line 172 of file generated_mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.22.2.5 subroutine `GENERATED_MESH_ROUTINES::GENERATED_MESH_INITIALISE`
`(TYPE(GENERATED_MESH_TYPE),pointer GENERATED_MESH,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Initialises a generated mesh.

Parameters:

GENERATED_MESH A pointer to the generated mesh to initialise
ERR The error code
ERROR The error string

Definition at line 197 of file generated_mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.22.2.6 subroutine GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET
(TYPE(GENERATED_MESH_TYPE),pointer *GENERATED_MESH*,
INTEGER(INTG),intent(in) *GENERATED_TYPE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Sets/changes the type of a generated mesh.

Parameters:

GENERATED_MESH A pointer to the generated mesh to set the type of

GENERATED_TYPE The type of mesh to generate

See also:

[GENERATED_MESH_ROUTINES::GeneratedMeshTypes](#),[GENERATED_MESH_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 229 of file generated_mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.23 INPUT_OUTPUT Namespace Reference

This module handles all formating and input and output.

Classes

- interface [WRITE_STRING](#)
Write a string to a given output stream.
- interface [WRITE_STRING_VALUE](#)
Write a string followed by a value to a given output stream.
- interface [WRITE_STRING_TWO_VALUE](#)
Write a string, value, string then a value to a given output stream.
- interface [WRITE_STRING_FMT_VALUE](#)
Write a string followed by a value formatted in a particular way to a specified output stream.
- interface [WRITE_STRING_FMT_TWO_VALUE](#)
Write a string, value, string then a value with the values formatted in a particular way to a given output stream.
- interface [WRITE_STRING_VECTOR](#)
Write a string followed by a vector to a specified output stream.
- interface [WRITE_STRING_IDX_VECTOR](#)
Write a string followed by a indexed vector to a specified output stream.
- interface [WRITE_STRING_MATRIX](#)
Write a string followed by a matrix to a specified output stream.

Functions

- subroutine [WRITE_STRING_C](#) (ID, STRING, ERR, ERROR,*)
Writes the character STRING to the given output stream specified by ID.
- subroutine [WRITE_STRING_VS](#) (ID, STRING, ERR, ERROR,*)
Writes the varying string STRING to the given output stream specified by ID.
- subroutine [WRITE_STRING_VALUE_C](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_VALUE_DP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_VALUE_INTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

- subroutine [WRITE_STRING_VALUE_LINTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_VALUE_L](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_VALUE_SP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_VALUE_VS](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [WRITE_STRING_TWO_VALUE_C_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [WRITE_STRING_TWO_VALUE_C_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [WRITE_STRING_TWO_VALUE_C_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [WRITE_STRING_TWO_VALUE_C_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [WRITE_STRING_TWO_VALUE_C_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [WRITE_STRING_TWO_VALUE_C_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_DP_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_INTG_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_L_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_SP_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_TWO_VALUE_VS_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [WRITE_STRING_FMT_VALUE_C](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine [WRITE_STRING_FMT_VALUE_DP](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine [WRITE_STRING_FMT_VALUE_INTG](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine [WRITE_STRING_FMT_VALUE_LINTG](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine [WRITE_STRING_FMT_VALUE_L](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine [WRITE_STRING_FMT_VALUE_SP](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine **WRITE_STRING_FMT_VALUE_VS** (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given logical VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [WRITE_STRING_IDX](#) (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed double precision VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine [WRITE_STRING_IDX](#) (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine [WRITE_STRING_IDX](#) (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine [WRITE_STRING_IDX](#) (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the

number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `WRITE_STRING_IDX` (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `WRITE_STRING_MATRIX_DP` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRS,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `WRITE_STRING_MATRIX_INTG` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FI ST,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `WRITE_STRING_MATRIX_LINTG` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_F RST,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)
- subroutine `WRITE_STRING_MATRIX_L` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRST &NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `WRITE_STRING_MATRIX_SP` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRS,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Variables

- INTEGER(INTG), parameter `WRITE_STRING_MATRIX_NAME_ONLY` = 1

Write the matrix name with out any indices.

- INTEGER(INTG), parameter `WRITE_STRING_MATRIX_NAME_AND_INDICES` = 2

Write the matrix name together with the matrix indices.

6.23.1 Detailed Description

This module handles all formating and input and output.

6.23.2 Function Documentation

**6.23.2.1 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 INTEGER(INTG),intent(in) *FIRST_IDX*, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) *LAST_IDX*, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) *NUMBER_REPEAT*,
 REAL(SP),dimension(:),intent(in) *VECTOR*, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) *REPEAT_FORMAT*, &,intent(out)
ERR, INTEGER(INTG),intent(out) *error*, *)**

Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_IDX The first index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

LAST_IDX The last index of the vector to output

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

Definition at line 3380 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.2 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 INTEGER(INTG),intent(in) *FIRST_IDX*, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) *LAST_IDX*, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) *NUMBER_REPEAT*,
 LOGICAL,dimension(:),intent(in) *VECTOR*, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) *REPEAT_FORMAT*, &,intent(out)
ERR, INTEGER(INTG),intent(out) *error*, *)**

Writes the given logical VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_IDX The first index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

LAST_IDX The last index of the vector to output

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

Definition at line 3335 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.3 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
INTEGER(INTG),intent(in) FIRST_IDX, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) LAST_IDX, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) NUMBER_REPEAT,
INTEGER(LINTG),dimension(:,intent(in) VECTOR, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, &,intent(out)
ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_IDX The first index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

LAST_IDX The last index of the vector to output

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

Definition at line 3290 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.4 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
INTEGER(INTG),intent(in) FIRST_IDX, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) LAST_IDX, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) NUMBER_REPEAT,
INTEGER(INTG),dimension(:,),intent(in) VECTOR, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, &,intent(out)
ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_IDX The first index of the vector to output

DELTA The delta increment to be used when outputing the first through to the last vector index

LAST_IDX The last index of the vector to output

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

Definition at line 3245 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.5 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
INTEGER(INTG),intent(in) *FIRST_IDX*, INTEGER(INTG),intent(in)
DELTA, INTEGER(INTG),intent(in) *LAST_IDX*, INTEGER(INTG),intent(in)
NUMBER_FIRST, INTEGER(INTG),intent(in) *NUMBER_REPEAT*,
REAL(DP),dimension(:,intent(in) *VECTOR*, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) *REPEAT_FORMAT*, &,intent(out)
ERR, INTEGER(INTG),intent(out) *error*, *)**

Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_IDX The first index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

LAST_IDX The last index of the vector to output

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

Definition at line 3200 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.6 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_-
STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*),intent(in) *SECOND_-
STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*,
CHARACTER(LEN=*),intent(in) *SECOND_FORMAT*, &,intent(out) *ERR*,
INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 3166 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.7 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_FORMAT*, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 3131 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.8 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*=*),intent(in) *FIRST_STRING*, TYPE(VARYING_-
 STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*=*),intent(in) *SECOND_STRING*,
 LOGICAL,intent(in) *SECOND_VALUE*, CHARACTER(LEN=*=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 3097 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.9 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*=*),intent(in) *FIRST_STRING*, TYPE(VARYING_-
 STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*=*),intent(in)
FIRST_FORMAT, CHARACTER(LEN=*=*),intent(in) *SECOND_STRING*,
 REAL(DP),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 3027 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.10 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, TYPE(VARYING_
 STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*)intent(in)
FIRST_FORMAT, CHARACTER(LEN=*)intent(in) *SECOND_STRING*,
 CHARACTER(LEN=*)intent(in) *SECOND_VALUE*, CHARACTER(LEN=*)intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the *FIRST_STRING* followed by a formatted varying string *FIRST_VALUE* and the *SECOND_STRING* followed by a formatted character *SECOND_VALUE* to the given output stream specified by *ID*. *FIRST_FORMAT* is used to format the first value and *SECOND_FORMAT* is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2993 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.11 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*, TYPE(VARYING_
 STRING),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*)intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the *FIRST_STRING* followed by a formatted single precision *FIRST_VALUE* and the *SECOND_STRING* followed by a formatted single precision *SECOND_VALUE* to the given output stream specified by *ID*. *FIRST_FORMAT* is used to format the first value and *SECOND_FORMAT* is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2958 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.12 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the **FIRST_STRING** followed by a formatted single precision **FIRST_VALUE** and the **SECOND_STRING** followed by a formatted single precision **SECOND_VALUE** to the given output stream specified by **ID**. **FIRST_FORMAT** is used to format the first value and **SECOND_FORMAT** is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2919 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.13 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2883 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.14 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*, REAL(DP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2805 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.15 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_
FORMAT, CHARACTER(LEN=*),intent(in) SECOND_STRING,
CHARACTER(LEN=*),intent(in) SECOND_VALUE, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2770 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.16 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, TYPE(VARYING_-
STRING),intent(in) SECOND_VALUE, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2735 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.17 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2698 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.18 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2660 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.19 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2623 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.20 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*, REAL(DP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the *FIRST_STRING* followed by a formatted logical *FIRST_VALUE* and the the *SECOND_STRING* followed by a formatted double precision *SECOND_VALUE* to the given output stream specified by *ID*. *FIRST_FORMAT* is used to format the first value and *SECOND_FORMAT* is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2584 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.21 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*)intent(in) *FIRST_STRING*, LOGICAL,intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*)intent(in) *SECOND_STRING*,
 CHARACTER(LEN=*)intent(in) *SECOND_VALUE*, CHARACTER(LEN=*)intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the *FIRST_STRING* followed by a formatted logical *FIRST_VALUE* and the the *SECOND_STRING* followed by a formatted character *SECOND_VALUE* to the given output stream specified by *ID*. *FIRST_FORMAT* is used to format the first value and *SECOND_FORMAT* is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2549 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.22 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, INTEGER(INTG),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2437 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.23 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
CHARACTER(LEN=*),intent(in) *SECOND_STRING*,
CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2324 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.24 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_-
STRING),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2289 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.25 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
 CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
 FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
 SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
 &,intent(out) ERR, INTEGER(INTG),intent(out) error, *)**

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2250 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.26 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
 CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
 FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
 CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in)
 SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
 &,intent(out) ERR, INTEGER(INTG),intent(out) error, *)**

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2212 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.27 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(DP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the **FIRST_STRING** followed by a formatted double precision **FIRST_VALUE** and the **SECOND_STRING** followed by a formatted double precision **SECOND_VALUE** to the given output stream specified by **ID**. **FIRST_FORMAT** is used to format the first value and **SECOND_FORMAT** is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2134 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.28 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*,
 CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2099 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.29 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_-
STRING),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2065 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.30 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(SP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 2030 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.31 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, LOGICAL,intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 1996 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.32 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*)intent(in) FIRST_STRING, CHARACTER(LEN=*)intent(in)
FIRST_VALUE, CHARACTER(LEN=*)intent(in) FIRST_FORMAT,
CHARACTER(LEN=*)intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
SECOND_VALUE, CHARACTER(LEN=*)intent(in) SECOND_FORMAT,
&,intent(out) ERR, INTEGER(INTG),intent(out) error, *)
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 1961 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.33 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in)
SECOND_VALUE, CHARACTER(LEN=*),intent(in) *SECOND_FORMAT*,
 &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 1926 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

**6.23.2.34 subroutine INPUT_OUTPUT::WR (INTEGER(INTG),intent(in) *ID*,
 CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*,
 CHARACTER(LEN=*),intent(in) *SECOND_STRING*,
 CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_FORMAT, &,intent(out) *ERR*, INTEGER(INTG),intent(out) *error*, *)**

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

SECOND_FORMAT The format string to be used to format the second value

Definition at line 1892 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.35 subroutine INPUT_OUTPUT::WRITE_STRING_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the character *STRING* to the given output stream specified by *ID*.

Parameters:

ID The ID of the output stream to write to

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

STRING The string to write

ERR The error code

ERROR The error string

Definition at line 221 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.36 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, &,intent(in) *SECOND_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the *FIRST_STRING* followed by a formatted varying string *FIRST_VALUE* and the *SECOND_STRING* followed by a formatted integer *SECOND_VALUE* to the given output stream specified by *ID*. *FIRST_FORMAT* is used to format the first value and *SECOND_FORMAT* is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 3062 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.37 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in)
ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in)
FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
SECOND_VALUE, &,intent(in) SECOND_FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2844 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

**6.23.2.38 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_-
STRING),intent(in) *SECOND_VALUE*, &,intent(in) *SECOND_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2514 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.39 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, &,intent(in) *SECOND_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2475 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.40 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in)
  ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, INTEGER(INTG),intent(in)
  FIRST_VALUE, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
  CHARACTER(LEN=*),intent(in) SECOND_STRING, INTEGER(INTG),intent(in)
  SECOND_VALUE, &,intent(in) SECOND_FORMAT, INTEGER(INTG),intent(out)
  ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2398 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.41 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, &,intent(in) *SECOND_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2359 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.42 subroutine INPUT_OUTPUT::WRITE_STRING_FMT (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, &,intent(in) *SECOND_FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

FIRST_FORMAT The format string to be used to format the first value

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 2173 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.43 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, CHARACTER(LEN=*),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1680 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.44 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DP
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(DP),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1710 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.45 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG
(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
INTEGER(INTG),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_
STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1741 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.46 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L
(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
LOGICAL,intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1801 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.47 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_LINTG
(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
INTEGER(LINTG),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_
STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1771 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.48 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP
(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
REAL(SP),intent(in) VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***) [private]**

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1831 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.49 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
   FIRST_STRING, TYPE(VARYING_STRING),intent(in) VALUE,
   CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out)
   ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1862 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.50 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) *ID*,
 INTEGER(INTG),intent(in) *NUM_INDICES*, INTEGER(INTG),dimension(*num_-
 indices*),intent(in) *INDICES*, INTEGER(INTG),intent(in) *DELTA*,
 INTEGER(INTG),intent(in) *NUMBER_FIRST*, INTEGER(INTG),intent(in)
NUMBER_REPEAT, REAL(SP),dimension(:),intent(in) *VECTOR*,
 CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, &,intent(in) *REPEAT_FORMAT*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

NUM_INDICES The number of indices of the vector to output

INDICES INDICES(i). The i'th index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

ERR The error code

ERROR The error string

Definition at line 3610 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.51 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) *ID*,
 INTEGER(INTG),intent(in) *NUM_INDICES*, INTEGER(INTG),dimension(*num_-
 indices*),intent(in) *INDICES*, INTEGER(INTG),intent(in) *DELTA*,
 INTEGER(INTG),intent(in) *NUMBER_FIRST*, INTEGER(INTG),intent(in)
NUMBER_REPEAT, LOGICAL,dimension(:),intent(in) *VECTOR*,
 CHARACTER(LEN=*),intent(in) *FIRST_FORMAT*, &,intent(in) *REPEAT_FORMAT*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT

is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

NUM_INDICES The number of indices of the vector to output

INDICES INDICES(i). The i'th index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

ERR The error code

ERROR The error string

Definition at line 3565 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.52 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,
      INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_indices),intent(in)
      INDICES, INTEGER(INTG),intent(in) DELTA,
      INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)
      NUMBER_REPEAT, INTEGER(LINTG),dimension(:),intent(in) VECTOR,
      CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,
      INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
      *) [private]
```

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

NUM_INDICES The number of indices of the vector to output

INDICES INDICES(i). The i'th index of the vector to output
DELTA The delta increment to be used when outputting the first through to the last vector index
NUMBER_FIRST The number of vector elements to be output on the first line
NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines
VECTOR The vector to be output
FIRST_FORMAT The format string to be used for the first line of output
ERR The error code
ERROR The error string

Definition at line 3515 of file input_output.f90.

References BASE_ROUTINES::ERRORS(), BASE_ROUTINES::OP_STRING, and BASE_ROUTINES::WRITE_STR().

Here is the call graph for this function:

```
6.23.2.53 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,
      INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_-
      indices),intent(in) INDICES, INTEGER(INTG),intent(in) DELTA,
      INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)
      NUMBER_REPEAT, INTEGER(INTG),dimension(:,intent(in) VECTOR,
      CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,
      INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
      *) [private]
```

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

NUM_INDICES The number of indices of the vector to output

INDICES INDICES(i). The i'th index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

ERR The error code

ERROR The error string

Definition at line 3470 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.54 subroutine INPUT_OUTPUT::WRITE_STRING_IDX (INTEGER(INTG),intent(in) ID,  
          INTEGER(INTG),intent(in) NUM_INDICES, INTEGER(INTG),dimension(num_-  
          indices),intent(in) INDICES, INTEGER(INTG),intent(in) DELTA,  
          INTEGER(INTG),intent(in) NUMBER_FIRST, INTEGER(INTG),intent(in)  
          NUMBER_REPEAT, REAL(DP),dimension(:),intent(in) VECTOR,  
          CHARACTER(LEN=*),intent(in) FIRST_FORMAT, &,intent(in) REPEAT_FORMAT,  
          INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,  
          *) [private]
```

Writes the given indexed double precision VECTOR to the given output stream specified by ID. *NUM_INDICES* is the number of indices and *INDICES*(i) contain the indices of the vector to write. The *FIRST_FORMAT* is the format initially used, followed by the *REPEAT_FORMAT* which is repeated as many times as necessary. *NUMBER_FIRST* is the number of data items in the *FIRST_FORMAT* and *NUMBER_REPEAT* is the number of data items in the *REPEAT_FORMAT*. *DELTA* is the number of actual indices to skip for each index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

NUM_INDICES The number of indices of the vector to output

INDICES *INDICES*(i). The i'th index of the vector to output

DELTA The delta increment to be used when outputting the first through to the last vector index

NUMBER_FIRST The number of vector elements to be output on the first line

NUMBER_REPEAT The number of vector elements to be output on the second and subsequently repeated lines

VECTOR The vector to be output

FIRST_FORMAT The format string to be used for the first line of output

ERR The error code

ERROR The error string

Definition at line 3425 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.55 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_DP

```
(INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
LAST_COLUMN, NUMBER_FIRS, &,intent(in) NUMBER_REPEAT,
INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3655 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, `BASE_ROUTINES::WRITE_STR()`, `WRITE_STRING_MATRIX_NAME_AND_INDICES`, and `WRITE_STRING_MATRIX_NAME_ONLY`.

Here is the call graph for this function:

```
6.23.2.56 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG
  (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
  INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
  LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
  INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
  LAST_COLUMN, NUMBER_FIRST, &,intent(in) NUMBER_REPEAT,
  INTEGER(INTG),dimension(:, :, :),intent(in) matrix, INTEGER(INTG),intent(in)
  INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
  MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
  ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
  CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
  ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat MATRIX_NAME_FORMAT](#) The format string to be used to format the matrix name
[INPUT_OUTPUT::MatrixNameIndexFormat ROW_INDEX_FORMAT](#) The format string to be used to format the row indices
[INPUT_OUTPUT::MatrixNameIndexFormat FIRST_FORMAT](#) The format string to be used for the first line of output
[INPUT_OUTPUT::MatrixNameIndexFormat REPEAT_FORMAT](#) The format type to be used for the second and subsequently repeated lines of output
[ERR](#) The error code
[ERROR](#) The error string

Definition at line 3720 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), [BASE_ROUTINES::WRITE_STR\(\)](#), [WRITE_STRING_MATRIX_NAME_AND_INDICES](#), and [WRITE_STRING_MATRIX_NAME_ONLY](#).

Here is the call graph for this function:

```
6.23.2.57 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_L
  (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
   INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
   LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
   INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
   LAST_COLUMN, NUMBER_FIRST &,intent(in) NUMBER_REPEAT,
   INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
   INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
   MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
   ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
   CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
   ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3857 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, `BASE_ROUTINES::WRITE_STR()`, `WRITE_STRING_MATRIX_NAME_AND_INDICES`, and `WRITE_STRING_MATRIX_NAME_ONLY`.

Here is the call graph for this function:

```
6.23.2.58 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG
  (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
   INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
   LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
   INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
   LAST_COLUMN, NUMBER_F_RST, &,intent(in) NUMBER_REPEAT,
   INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
   INDEX_FORMAT_TYPE, CHARACTER(LEN=*),intent(in)
   MATRIX_NAME_FORMAT, CHARACTER(LEN=*),intent(in)
   ROW_INDEX_FORMAT, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
   CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
   ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given long integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the

extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3792 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), [BASE_ROUTINES::WRITE_STR\(\)](#), [WRITE_STRING_MATRIX_NAME_AND_INDICES](#), and [WRITE_STRING_MATRIX_NAME_ONLY](#).

Here is the call graph for this function:

```
6.23.2.59 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX_SP
  (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in) FIRST_ROW,
   INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
   LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
   INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
   LAST_COLUMN, NUMBER_FIRS, &,intent(in) NUMBER_REPEAT,
   INTEGER(INTG),dimension(:, :, :),intent(in) matrix, INTEGER(INTG),intent(in)
   INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
   MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
   ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
   CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
   ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_-

STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3922 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), [BASE_ROUTINES::WRITE_STR\(\)](#), [WRITE_STRING_MATRIX_NAME_AND_INDICES](#), and [WRITE_STRING_MATRIX_NAME_ONLY](#).

Here is the call graph for this function:

6.23.2.60 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_C
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 480 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.61 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_DP
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, REAL(DP),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 511 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.62 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 543 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.63 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, LOGICAL,intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 575 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.64 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_SP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   CHARACTER(LEN=*),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, REAL(SP),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 606 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.65 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 638 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.66 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_C
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 669 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.67 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
   STRING, REAL(DP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 701 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.68 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 737 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.69 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_-
STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 773 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.70 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_-
STRING, REAL(SP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output

stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 808 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.71 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   REAL(DP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the **FIRST_STRING** followed by a formatted double precision **FIRST_VALUE** and the **SECOND_STRING** followed by a formatted single precision **SECOND_VALUE** to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 844 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.72 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 876 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.73 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, REAL(DP),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 908 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.74 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 944 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.75 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, LOGICAL,intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 980 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.76 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   INTEGER(INTG),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, REAL(SP),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1015 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.77 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1051 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.78 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1083 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.79 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
   STRING, REAL(DP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the `FIRST_STRING` followed by a formatted logical `FIRST_VALUE` and the `SECOND_STRING` followed by a formatted double precision `SECOND_VALUE` to the given output stream specified by `ID`. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1115 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.80 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the `FIRST_STRING` followed by a formatted logical `FIRST_VALUE` and the `SECOND_STRING` followed by a formatted integer `SECOND_VALUE` to the given output stream specified by `ID`. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1151 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.81 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
 STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1185 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.82 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
 LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
 STRING, REAL(SP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified

by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1220 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

```
6.23.2.83 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   LOGICAL,intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the **FIRST_STRING** followed by a formatted logical **FIRST_VALUE** and the the **SECOND_STRING** followed by a formatted single precision **SECOND_VALUE** to the given output stream specified by *ID*. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1254 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.84 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in)
SECOND_STRING, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Writes the *FIRST_STRING* followed by a formatted single precision *FIRST_VALUE* and the *SECOND_STRING* followed by a formatted character *SECOND_VALUE* to the given output stream specified by *ID*. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1286 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.85 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP
 (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
 REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*,
 REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Writes the *FIRST_STRING* followed by a formatted single precision *FIRST_VALUE* and the *SECOND_STRING* followed by a formatted double precision *SECOND_VALUE* to the given output stream specified by *ID*. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1318 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.86 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   REAL(SP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
   SECOND_STRING, INTEGER(INTG),intent(in) SECOND_VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *) [private]
```

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1354 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.87 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
   REAL(SP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
   -STRING, LOGICAL,intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1390 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.88 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP
(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,
REAL(SP),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in) SECOND_
STRING, REAL(SP),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1423 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.89 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS
 (**INTEGER(INTG),intent(in) ID**, **CHARACTER(LEN=*)**,**intent(in) FIRST_STRING**,
REAL(SP),intent(in) FIRST_VALUE, **CHARACTER(LEN=*)**,**intent(in) SECOND_STRING**, **TYPE(VARYING_STRING)**,**intent(in) SECOND_VALUE**,
INTEGER(INTG),intent(out) ERR, **TYPE(VARYING_STRING)**,**intent(out) ERROR**,
 \ast) [private]

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1459 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.90 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_C
 (**INTEGER(INTG),intent(in) ID**, **CHARACTER(LEN=*)**,**intent(in) FIRST_STRING**, **TYPE(VARYING_STRING)**,**intent(in) FIRST_VALUE**, **CHARACTER(LEN=*)**,**intent(in) SECOND_STRING**, **CHARACTER(LEN=*)**,**intent(in) SECOND_VALUE**, **INTEGER(INTG),intent(out) ERR**, **TYPE(VARYING_STRING)**,**intent(out) ERROR**, \ast) [private]

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1491 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.91 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
   FIRST_STRING, TYPE(VARYING_STRING),intent(in) FIRST_VALUE,
   CHARACTER(LEN=*),intent(in) SECOND_STRING, REAL(DP),intent(in) SECOND_
   VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
   ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1522 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.92 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)
   FIRST_STRING, TYPE(VARYING_STRING),intent(in) FIRST_
   VALUE, CHARACTER(LEN=*),intent(in) SECOND_STRING,
   INTEGER(INTG),intent(in) SECOND_VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1554 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.93 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L
(INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in)
***FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*,**
CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
***ERROR*, *) [private]**

Writes the **FIRST_STRING** followed by a formatted varying string **FIRST_VALUE** and the the **SECOND_STRING** followed by a formatted logical **SECOND_VALUE** to the given output stream specified by **ID**. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1586 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [STRINGS::LOGICAL_TO_VSTRING\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.94 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP
 (**INTEGER(INTG),intent(in)** *ID*, **CHARACTER(LEN=*)**,**intent(in)**
FIRST_STRING, **TYPE(VARYING_STRING)**,**intent(in)** *FIRST_VALUE*,
CHARACTER(LEN=*),**intent(in)** *SECOND_STRING*, **REAL(SP)**,**intent(in)** *SECOND_-*
VALUE, **INTEGER(INTG)**,**intent(out)** *ERR*, **TYPE(VARYING_STRING)**,**intent(out)**
ERROR, *) [private]

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1617 of file input_output.f90.

References [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::OP_STRING](#), and [BASE_ROUTINES::WRITE_STR\(\)](#).

Here is the call graph for this function:

6.23.2.95 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS
 (**INTEGER(INTG),intent(in)** *ID*, **CHARACTER(LEN=*)**,**intent(in)**
FIRST_STRING, **TYPE(VARYING_STRING)**,**intent(in)** *FIRST_VALUE*,
CHARACTER(LEN=*),**intent(in)** *SECOND_STRING*, **TYPE(VARYING_-**
STRING),**intent(in)** *SECOND_VALUE*, **INTEGER(INTG)**,**intent(out)** *ERR*,
TYPE(VARYING_STRING),**intent(out)** *ERROR*, *) [private]

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1649 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.96 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_C
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`
`CHARACTER(LEN=*),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 273 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.97 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_DP
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in)`
`FIRST_STRING, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 302 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.98 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_INTG
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`
`INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 332 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.23.2.99 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_L
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) FIRST_STRING,`
`LOGICAL,intent(in) VALUE, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 392 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `STRINGS::LOGICAL_TO_VSTRING()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.100 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*)intent(in) FIRST_STRING,
   INTEGER(LINTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 362 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.101 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_SP
  (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*)intent(in)
   FIRST_STRING, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 421 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.102 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE_VS
    (INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*) intent(in) FIRST_STRING,
     TYPE(VARYING_STRING),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
     TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 451 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

```
6.23.2.103 subroutine INPUT_OUTPUT::WRITE_STRING_VS (INTEGER(INTG),intent(in) ID,
    TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
    TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Writes the varying string STRING to the given output stream specified by ID.

Parameters:

ID The ID of the output stream to write to

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

STRING The string to write

ERR The error code

ERROR The error string

Definition at line 247 of file input_output.f90.

References `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::OP_STRING`, and `BASE_ROUTINES::WRITE_STR()`.

Here is the call graph for this function:

6.24 ISO_VARYING_STRING Namespace Reference

This module provides an iso_varying_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).

Classes

- struct `VARYING_STRING`
- interface `assignment(=)`
- interface `adjustr`
- interface `char`
- interface `iachar`
- interface `ichar`
- interface `index`
- interface `len`
- interface `len_trim`
- interface `lge`
- interface `lgt`
- interface `lle`
- interface `llt`
- interface `repeat`
- interface `scan`
- interface `trim`
- interface `verify`
- interface `var_str`
- interface `get`
- interface `put`
- interface `put_line`
- interface `extract`
- interface `insert`
- interface `remove`
- interface `replace`
- interface `split`

Functions

- `integer len_(string)`
- subroutine `op_assign_CH_VS` (`var, exp`)
- subroutine `op_assign_VS_CH` (`var, exp`)
- type(varying_string) `op_concat_VS_VS` (`string_a, string_b`)
- type(varying_string) `op_concat_CH_VS` (`string_a, string_b`)
- type(varying_string) `op_concat_VS_CH` (`string_a, string_b`)
- logical `op_eq_VS_VS` (`string_a, string_b`)
- logical `op_eq_CH_VS` (`string_a, string_b`)
- logical `op_eq_VS_CH` (`string_a, string_b`)
- logical `op_ne_VS_VS` (`string_a, string_b`)
- logical `op_ne_CH_VS` (`string_a, string_b`)
- logical `op_ne_VS_CH` (`string_a, string_b`)
- logical `op_lt_VS_VS` (`string_a, string_b`)

- `logical op_lt_CH_VS (string_a, string_b)`
- `logical op_lt_VS_CH (string_a, string_b)`
- `logical op_le_VS_VS (string_a, string_b)`
- `logical op_le_CH_VS (string_a, string_b)`
- `logical op_le_VS_CH (string_a, string_b)`
- `logical op_ge_VS_VS (string_a, string_b)`
- `logical op_ge_CH_VS (string_a, string_b)`
- `logical op_ge_VS_CH (string_a, string_b)`
- `logical op_gt_VS_VS (string_a, string_b)`
- `logical op_gt_CH_VS (string_a, string_b)`
- `logical op_gt_VS_CH (string_a, string_b)`
- `type(varying_string) adjustl_ (string)`
- `type(varying_string) adjustr_ (string)`
- `function char_auto (string)`
- `character(LEN=length) char_fixed (string, length)`
- `integer iachar_ (c)`
- `integer ichar_ (c)`
- `integer index_VS_VS (string, substring, back)`
- `integer index_CH_VS (string, substring, back)`
- `integer index_VS_CH (string, substring, back)`
- `integer len_trim_ (string)`
- `logical lge_VS_VS (string_a, string_b)`
- `logical lge_CH_VS (string_a, string_b)`
- `logical lge_VS_CH (string_a, string_b)`
- `logical lgt_VS_VS (string_a, string_b)`
- `logical lgt_CH_VS (string_a, string_b)`
- `logical lgt_VS_CH (string_a, string_b)`
- `logical lle_VS_VS (string_a, string_b)`
- `logical lle_CH_VS (string_a, string_b)`
- `logical lle_VS_CH (string_a, string_b)`
- `logical llt_VS_VS (string_a, string_b)`
- `logical llt_CH_VS (string_a, string_b)`
- `logical llt_VS_CH (string_a, string_b)`
- `type(varying_string) repeat_ (string, ncopies)`
- `integer scan_VS_VS (string, set, back)`
- `integer scan_CH_VS (string, set, back)`
- `integer scan_VS_CH (string, set, back)`
- `type(varying_string) trim_ (string)`
- `integer verify_VS_VS (string, set, back)`
- `integer verify_CH_VS (string, set, back)`
- `integer verify_VS_CH (string, set, back)`
- `type(varying_string) var_str_ (char)`
- `subroutine get_ (string, maxlen, iostat)`
- `subroutine get_unit (unit, string, maxlen, iostat)`
- `subroutine get_set_VS (string, set, separator, maxlen, iostat)`
- `subroutine get_set_CH (string, set, separator, maxlen, iostat)`
- `subroutine get_unit_set_VS (unit, string, set, separator, maxlen, iostat)`
- `subroutine get_unit_set_CH (unit, string, set, separator, maxlen, iostat)`
- `subroutine put_VS (string, iostat)`
- `subroutine put_CH (string, iostat)`

- subroutine `put_unit_VS` (unit, string, iostat)
- subroutine `put_unit_CH` (unit, string, iostat)
- subroutine `put_line_VS` (string, iostat)
- subroutine `put_line_CH` (string, iostat)
- subroutine `put_line_unit_VS` (unit, string, iostat)
- subroutine `put_line_unit_CH` (unit, string, iostat)
- type(varying_string) `extract_VS` (string, start, finish)
- type(varying_string) `extract_CH` (string, start, finish)
- type(varying_string) `insert_VS_VS` (string, start, substring)
- type(varying_string) `insert_CH_VS` (string, start, substring)
- type(varying_string) `insert_VS_CH` (string, start, substring)
- type(varying_string) `insert_CH_CH` (string, start, substring)
- TYPE(varying_string) `remove_VS` (string, start, finish)
- type(varying_string) `remove_CH` (string, start, finish)
- type(varying_string) `replace_VS_VS_auto` (string, start, substring)
- type(varying_string) `replace_CH_VS_auto` (string, start, substring)
- type(varying_string) `replace_VS_CH_auto` (string, start, substring)
- type(varying_string) `replace_CH_CH_auto` (string, start, substring)
- type(varying_string) `replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying_string) `replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying_string) `replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying_string) `replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying_string) `replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_CH_CH_target` (string, target, substring, every, back)
- subroutine `split_VS` (string, word, set, separator, back)
- subroutine `split_CH` (string, word, set, separator, back)

Variables

- `integer`, parameter `GET_BUFFER_LEN` = 256

6.24.1 Detailed Description

This module provides an iso_varying_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).

6.24.2 Function Documentation

6.24.2.1 type(varying_string) ISO_VARYING_STRING::adjustl_ (type(varying_string),intent(in) string) [private]

Definition at line 858 of file iso_varying_string.f90.

6.24.2.2 type(varying_string) ISO_VARYING_STRING::adjustr_ (type(varying_string),intent(in) string) [private]

Definition at line 875 of file iso_varying_string.f90.

6.24.2.3 function ISO_VARYING_STRING::char_auto (type(varying_string),intent(in) string) [private]

Definition at line 892 of file iso_varying_string.f90.

**6.24.2.4 character(LEN=length) ISO_VARYING_STRING::char_fixed
(type(varying_string),intent(in) string, integer,intent(in) length) [private]**

Definition at line 914 of file iso_varying_string.f90.

**6.24.2.5 type(varying_string) ISO_VARYING_STRING::extract_CH
(character(LEN=*)intent(in) string, integer,intent(in),optional start,
integer,intent(in),optional finish) [private]**

Definition at line 1901 of file iso_varying_string.f90.

6.24.2.6 type(varying_string) ISO_VARYING_STRING::extract_VS (type(varying_string),intent(in) string, integer,intent(in),optional start, integer,intent(in),optional finish) [private]

Definition at line 1882 of file iso_varying_string.f90.

6.24.2.7 subroutine ISO_VARYING_STRING::get_ (type(varying_string),intent(out) string, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]

Definition at line 1455 of file iso_varying_string.f90.

References GET_BUFFER_LEN.

6.24.2.8 subroutine ISO_VARYING_STRING::get_set_CH (type(varying_string),intent(out) string, character(LEN=*)intent(in) set, type(varying_string),intent(out),optional separator, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]

Definition at line 1583 of file iso_varying_string.f90.

6.24.2.9 subroutine ISO_VARYING_STRING::get_set_VS (type(varying_string),intent(out) string, type(varying_string),intent(in) set, type(varying_string),intent(out),optional separator, integer,intent(in),optional maxlen, integer,intent(out),optional iostat) [private]

Definition at line 1562 of file iso_varying_string.f90.

6.24.2.10 subroutine ISO_VARYING_STRING::get_unit (integer,intent(in) *unit*, type(varying_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]

Definition at line 1508 of file iso_varying_string.f90.

References GET_BUFFER_LEN.

6.24.2.11 subroutine ISO_VARYING_STRING::get_unit_set_CH (integer,intent(in) *unit*, type(varying_string),intent(out) *string*, character(LEN=*),intent(in) *set*, type(varying_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]

Definition at line 1663 of file iso_varying_string.f90.

6.24.2.12 subroutine ISO_VARYING_STRING::get_unit_set_VS (integer,intent(in) *unit*, type(varying_string),intent(out) *string*, type(varying_string),intent(in) *set*, type(varying_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*) [private]

Definition at line 1641 of file iso_varying_string.f90.

6.24.2.13 integer ISO_VARYING_STRING::iachar_ (type(varying_string),intent(in) *c*) [private]

Definition at line 933 of file iso_varying_string.f90.

6.24.2.14 integer ISO_VARYING_STRING::ichar_ (type(varying_string),intent(in) *c*) [private]

Definition at line 951 of file iso_varying_string.f90.

6.24.2.15 integer ISO_VARYING_STRING::index_CH_VS (character(LEN=*),intent(in) *string*, type(varying_string),intent(in) *substring*, logical,intent(in),optional *back*) [private]

Definition at line 989 of file iso_varying_string.f90.

6.24.2.16 integer ISO_VARYING_STRING::index_VS_CH (type(varying_string),intent(in) *string*, character(LEN=*),intent(in) *substring*, logical,intent(in),optional *back*) [private]

Definition at line 1009 of file iso_varying_string.f90.

6.24.2.17 integer ISO_VARYING_STRING::index_VS_VS (type(varying_string),intent(in) *string*, type(varying_string),intent(in) *substring*, logical,intent(in),optional *back*) [private]

Definition at line 969 of file iso_varying_string.f90.

6.24.2.18 `type(varying_string) ISO_VARYING_STRING::insert_CH_CH
(character(LEN=*),intent(in) string, integer,intent(in) start,
character(LEN=*),intent(in) substring)` [private]

Definition at line 1992 of file iso_varying_string.f90.

6.24.2.19 `type(varying_string) ISO_VARYING_STRING::insert_CH_VS
(character(LEN=*),intent(in) string, integer,intent(in) start,
type(varying_string),intent(in) substring)` [private]

Definition at line 1954 of file iso_varying_string.f90.

6.24.2.20 `type(varying_string) ISO_VARYING_STRING::insert_VS_CH (type(varying_-
string),intent(in) string, integer,intent(in) start, character(LEN=*),intent(in) substring)
[private]`

Definition at line 1973 of file iso_varying_string.f90.

6.24.2.21 `type(varying_string) ISO_VARYING_STRING::insert_VS_VS (type(varying_-
string),intent(in) string, integer,intent(in) start, type(varying_string),intent(in)
substring)` [private]

Definition at line 1935 of file iso_varying_string.f90.

6.24.2.22 `integer ISO_VARYING_STRING::len_ (type(varying_string),intent(in) string)
[private]`

Definition at line 398 of file iso_varying_string.f90.

6.24.2.23 `integer ISO_VARYING_STRING::len_trim_ (type(varying_string),intent(in) string)
[private]`

Definition at line 1029 of file iso_varying_string.f90.

6.24.2.24 `logical ISO_VARYING_STRING::lge_CH_VS (character(LEN=*),intent(in) string_a,
type(varying_string),intent(in) string_b)` [private]

Definition at line 1068 of file iso_varying_string.f90.

6.24.2.25 `logical ISO_VARYING_STRING::lge_VS_CH (type(varying_string),intent(in) string_a,
character(LEN=*),intent(in) string_b)` [private]

Definition at line 1087 of file iso_varying_string.f90.

6.24.2.26 `logical ISO_VARYING_STRING::lge_VS_VS (type(varying_string),intent(in) string_a,
type(varying_string),intent(in) string_b)` [private]

Definition at line 1050 of file iso_varying_string.f90.

6.24.2.27 logical ISO_VARYING_STRING::lgt_CH_VS (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1124 of file iso_varying_string.f90.

6.24.2.28 logical ISO_VARYING_STRING::lgt_VS_CH (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*) [private]

Definition at line 1143 of file iso_varying_string.f90.

6.24.2.29 logical ISO_VARYING_STRING::lgt_VS_VS (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1106 of file iso_varying_string.f90.

6.24.2.30 logical ISO_VARYING_STRING::lle_CH_VS (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1180 of file iso_varying_string.f90.

6.24.2.31 logical ISO_VARYING_STRING::lle_VS_CH (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*) [private]

Definition at line 1199 of file iso_varying_string.f90.

6.24.2.32 logical ISO_VARYING_STRING::lle_VS_VS (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1162 of file iso_varying_string.f90.

6.24.2.33 logical ISO_VARYING_STRING::llt_CH_VS (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1236 of file iso_varying_string.f90.

6.24.2.34 logical ISO_VARYING_STRING::llt_VS_CH (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*) [private]

Definition at line 1255 of file iso_varying_string.f90.

6.24.2.35 logical ISO_VARYING_STRING::llt_VS_VS (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*) [private]

Definition at line 1218 of file iso_varying_string.f90.

6.24.2.36 subroutine ISO_VARYING_STRING::op_assign_CH_VS
 (character(LEN=*),intent(out) *var*, type(varying_string),intent(in) *exp*) [private]

Definition at line 419 of file iso_varying_string.f90.

6.24.2.37 subroutine ISO_VARYING_STRING::op_assign_VS_CH
 (type(varying_string),intent(out) *var*, character(LEN=*),intent(in) *exp*) [private]

Definition at line 436 of file iso_varying_string.f90.

6.24.2.38 type(varying_string) ISO_VARYING_STRING::op_concat_CH_VS
 (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)
 [private]

Definition at line 484 of file iso_varying_string.f90.

References op_concat_VS_VS().

Here is the call graph for this function:

6.24.2.39 type(varying_string) ISO_VARYING_STRING::op_concat_VS_CH
 (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)
 [private]

Definition at line 503 of file iso_varying_string.f90.

References op_concat_VS_VS().

Here is the call graph for this function:

6.24.2.40 type(varying_string) ISO_VARYING_STRING::op_concat_VS_VS
 (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)
 [private]

Definition at line 453 of file iso_varying_string.f90.

Referenced by op_concat_CH_VS(), and op_concat_VS_CH().

Here is the caller graph for this function:

6.24.2.41 logical ISO_VARYING_STRING::op_eq_CH_VS (character(LEN=*),intent(in)
string_a, type(varying_string),intent(in) *string_b*) [private]

Definition at line 540 of file iso_varying_string.f90.

6.24.2.42 logical ISO_VARYING_STRING::op_eq_VS_CH (type(varying_string),intent(in)
string_a, character(LEN=*),intent(in) *string_b*) [private]

Definition at line 559 of file iso_varying_string.f90.

6.24.2.43 logical ISO_VARYING_STRING::op_eq_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 522 of file iso_varying_string.f90.

6.24.2.44 logical ISO_VARYING_STRING::op_ge_CH_VS (character(LEN=*),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 764 of file iso_varying_string.f90.

6.24.2.45 logical ISO_VARYING_STRING::op_ge_VS_CH (type(varying_string),intent(in) string_a, character(LEN=*),intent(in) string_b) [private]

Definition at line 783 of file iso_varying_string.f90.

6.24.2.46 logical ISO_VARYING_STRING::op_ge_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 746 of file iso_varying_string.f90.

6.24.2.47 logical ISO_VARYING_STRING::op_gt_CH_VS (character(LEN=*),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 820 of file iso_varying_string.f90.

6.24.2.48 logical ISO_VARYING_STRING::op_gt_VS_CH (type(varying_string),intent(in) string_a, character(LEN=*),intent(in) string_b) [private]

Definition at line 839 of file iso_varying_string.f90.

6.24.2.49 logical ISO_VARYING_STRING::op_gt_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 802 of file iso_varying_string.f90.

6.24.2.50 logical ISO_VARYING_STRING::op_le_CH_VS (character(LEN=*),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 708 of file iso_varying_string.f90.

6.24.2.51 logical ISO_VARYING_STRING::op_le_VS_CH (type(varying_string),intent(in) string_a, character(LEN=*),intent(in) string_b) [private]

Definition at line 727 of file iso_varying_string.f90.

6.24.2.52 logical ISO_VARYING_STRING::op_le_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 690 of file iso_varying_string.f90.

6.24.2.53 logical ISO_VARYING_STRING::op_lt_CH_VS (character(LEN=*),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 652 of file iso_varying_string.f90.

6.24.2.54 logical ISO_VARYING_STRING::op_lt_VS_CH (type(varying_string),intent(in) string_a, character(LEN=*),intent(in) string_b) [private]

Definition at line 671 of file iso_varying_string.f90.

6.24.2.55 logical ISO_VARYING_STRING::op_lt_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 634 of file iso_varying_string.f90.

6.24.2.56 logical ISO_VARYING_STRING::op_ne_CH_VS (character(LEN=*),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 596 of file iso_varying_string.f90.

6.24.2.57 logical ISO_VARYING_STRING::op_ne_VS_CH (type(varying_string),intent(in) string_a, character(LEN=*),intent(in) string_b) [private]

Definition at line 615 of file iso_varying_string.f90.

6.24.2.58 logical ISO_VARYING_STRING::op_ne_VS_VS (type(varying_string),intent(in) string_a, type(varying_string),intent(in) string_b) [private]

Definition at line 578 of file iso_varying_string.f90.

6.24.2.59 subroutine ISO_VARYING_STRING::put_CH (character(LEN=*),intent(in) string, integer,intent(out),optional iostat) [private]

Definition at line 1738 of file iso_varying_string.f90.

6.24.2.60 subroutine ISO_VARYING_STRING::put_line_CH (character(LEN=*),intent(in) string, integer,intent(out),optional iostat) [private]

Definition at line 1818 of file iso_varying_string.f90.

6.24.2.61 subroutine ISO_VARYING_STRING::put_line_unit_CH (integer,intent(in) *unit*, character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1859 of file iso_varying_string.f90.

6.24.2.62 subroutine ISO_VARYING_STRING::put_line_unit_VS (integer,intent(in) *unit*, type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1840 of file iso_varying_string.f90.

6.24.2.63 subroutine ISO_VARYING_STRING::put_line_VS (type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1800 of file iso_varying_string.f90.

6.24.2.64 subroutine ISO_VARYING_STRING::put_unit_CH (integer,intent(in) *unit*, character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1777 of file iso_varying_string.f90.

6.24.2.65 subroutine ISO_VARYING_STRING::put_unit_VS (integer,intent(in) *unit*, type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1758 of file iso_varying_string.f90.

6.24.2.66 subroutine ISO_VARYING_STRING::put_VS (type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*) [private]

Definition at line 1722 of file iso_varying_string.f90.

**6.24.2.67 type(varying_string) ISO_VARYING_STRING::remove_CH
(character(LEN=*),intent(in) *string*, integer,intent(in),optional *start*, integer,intent(in),optional *finish*) [private]**

Definition at line 2035 of file iso_varying_string.f90.

6.24.2.68 TYPE(varying_string) ISO_VARYING_STRING::remove_VS (type(varying_string),intent(in) *string*, integer,intent(in),optional *start*, integer,intent(in),optional *finish*) [private]

Definition at line 2016 of file iso_varying_string.f90.

6.24.2.69 type(varying_string) ISO_VARYING_STRING::repeat_ (type(varying_string),intent(in) *string*, integer,intent(in) *ncopies*) [private]

Definition at line 1274 of file iso_varying_string.f90.

6.24.2.70 type(varying_string) ISO_VARYING_STRING::replace_CH_CH_auto
 (character(LEN=*),intent(in) *string*, integer,intent(in) *start*,
 character(LEN=*),intent(in) *substring*) [private]

Definition at line 2133 of file iso_varying_string.f90.

6.24.2.71 type(varying_string) ISO_VARYING_STRING::replace_CH_CH_CH_target
 (character(LEN=*),intent(in) *string*, character(LEN=*),intent(in) *target*,
 character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*) [private]

Definition at line 2410 of file iso_varying_string.f90.

6.24.2.72 type(varying_string) ISO_VARYING_STRING::replace_CH_CH_fixed
 (character(LEN=*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
 character(LEN=*),intent(in) *substring*) [private]

Definition at line 2218 of file iso_varying_string.f90.

6.24.2.73 type(varying_string) ISO_VARYING_STRING::replace_CH_CH_VS_target
 (character(LEN=*),intent(in) *string*, character(LEN=*),intent(in) *target*,
 type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*) [private]

Definition at line 2318 of file iso_varying_string.f90.

6.24.2.74 type(varying_string) ISO_VARYING_STRING::replace_CH_VS_auto
 (character(LEN=*),intent(in) *string*, integer,intent(in) *start*,
 type(varying_string),intent(in) *substring*) [private]

Definition at line 2093 of file iso_varying_string.f90.

6.24.2.75 type(varying_string) ISO_VARYING_STRING::replace_CH_VS_CH_target
 (character(LEN=*),intent(in) *string*, type(varying_string),intent(in) *target*,
 character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*) [private]

Definition at line 2364 of file iso_varying_string.f90.

6.24.2.76 type(varying_string) ISO_VARYING_STRING::replace_CH_VS_fixed
 (character(LEN=*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
 type(varying_string),intent(in) *substring*) [private]

Definition at line 2176 of file iso_varying_string.f90.

**6.24.2.77 type(varying_string) ISO_VARYING_STRING::replace_CH_VS_VS_target
(character(LEN=*),intent(in) *string*, type(varying_string),intent(in) *target*,
type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*) [private]**

Definition at line 2272 of file iso_varying_string.f90.

**6.24.2.78 type(varying_string) ISO_VARYING_STRING::replace_VS_CH_auto (type(varying_-
string),intent(in) *string*, integer,intent(in) *start*, character(LEN=*),intent(in) *substring*)
[private]**

Definition at line 2113 of file iso_varying_string.f90.

**6.24.2.79 type(varying_string) ISO_VARYING_STRING::replace_VS_CH_CH_target
(type(varying_string),intent(in) *string*, character(LEN=*),intent(in) *target*,
character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*) [private]**

Definition at line 2387 of file iso_varying_string.f90.

**6.24.2.80 type(varying_string) ISO_VARYING_STRING::replace_VS_CH_fixed
(type(varying_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
character(LEN=*),intent(in) *substring*) [private]**

Definition at line 2197 of file iso_varying_string.f90.

**6.24.2.81 type(varying_string) ISO_VARYING_STRING::replace_VS_CH_VS_target
(type(varying_string),intent(in) *string*, character(LEN=*),intent(in) *target*,
type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*) [private]**

Definition at line 2295 of file iso_varying_string.f90.

**6.24.2.82 type(varying_string) ISO_VARYING_STRING::replace_VS_VS_auto (type(varying_-
string),intent(in) *string*, integer,intent(in) *start*, type(varying_string),intent(in)
substring) [private]**

Definition at line 2073 of file iso_varying_string.f90.

**6.24.2.83 type(varying_string) ISO_VARYING_STRING::replace_VS_VS_CH_target
(type(varying_string),intent(in) *string*, type(varying_string),intent(in) *target*,
character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*) [private]**

Definition at line 2341 of file iso_varying_string.f90.

6.24.2.84 `type(varying_string) ISO_VARYING_STRING::replace_VS_VS_fixed
(type(varying_string),intent(in) string, integer,intent(in) start, integer,intent(in) finish,
type(varying_string),intent(in) substring) [private]`

Definition at line 2153 of file iso_varying_string.f90.

6.24.2.85 `type(varying_string) ISO_VARYING_STRING::replace_VS_VS_VS_target
(type(varying_string),intent(in) string, type(varying_string),intent(in) target,
type(varying_string),intent(in) substring, logical,intent(in),optional every,
logical,intent(in),optional back) [private]`

Definition at line 2249 of file iso_varying_string.f90.

6.24.2.86 `integer ISO_VARYING_STRING::scan_CH_VS (character(LEN=*)intent(in) string,
type(varying_string),intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1312 of file iso_varying_string.f90.

6.24.2.87 `integer ISO_VARYING_STRING::scan_VS_CH (type(varying_string),intent(in) string,
character(LEN=*)intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1332 of file iso_varying_string.f90.

6.24.2.88 `integer ISO_VARYING_STRING::scan_VS_VS (type(varying_string),intent(in) string,
type(varying_string),intent(in) set, logical,intent(in),optional back) [private]`

Definition at line 1292 of file iso_varying_string.f90.

6.24.2.89 `subroutine ISO_VARYING_STRING::split_CH (type(varying_string),intent(inout)
string, type(varying_string),intent(out) word, character(LEN=*)intent(in) set,
type(varying_string),intent(out),optional separator, logical,intent(in),optional back)
[private]`

Definition at line 2514 of file iso_varying_string.f90.

Referenced by split_VS().

Here is the caller graph for this function:

6.24.2.90 `subroutine ISO_VARYING_STRING::split_VS (type(varying_string),intent(inout)
string, type(varying_string),intent(out) word, type(varying_string),intent(in) set,
type(varying_string),intent(out),optional separator, logical,intent(in),optional back)
[private]`

Definition at line 2494 of file iso_varying_string.f90.

References split_CH().

Here is the call graph for this function:

6.24.2.91 type(varying_string) ISO_VARYING_STRING::trim_ (type(varying_string),intent(in) string) [private]

Definition at line 1352 of file iso_varying_string.f90.

6.24.2.92 type(varying_string) ISO_VARYING_STRING::var_str_ (character(LEN=*)intent(in) char) [private]

Definition at line 1429 of file iso_varying_string.f90.

6.24.2.93 integer ISO_VARYING_STRING::verify_CH_VS (character(LEN=*)intent(in) string, type(varying_string),intent(in) set, logical,intent(in),optional back) [private]

Definition at line 1389 of file iso_varying_string.f90.

6.24.2.94 integer ISO_VARYING_STRING::verify_VS_CH (type(varying_string),intent(in) string, character(LEN=*)intent(in) set, logical,intent(in),optional back) [private]

Definition at line 1409 of file iso_varying_string.f90.

6.24.2.95 integer ISO_VARYING_STRING::verify_VS_VS (type(varying_string),intent(in) string, type(varying_string),intent(in) set, logical,intent(in),optional back) [private]

Definition at line 1369 of file iso_varying_string.f90.

6.24.3 Variable Documentation

6.24.3.1 integer,parameter ISO_VARYING_STRING::GET_BUFFER_LEN = 256

Definition at line 54 of file iso_varying_string.f90.

Referenced by get_(), and get_unit().

6.25 KINDS Namespace Reference

This module contains all kind definitions.

Variables

- INTEGER, parameter **INTG** = SELECTED_INT_KIND(9)
Standard integer kind.
- INTEGER, parameter **SINTG** = SELECTED_INT_KIND(4)
Short integer kind.
- INTEGER, parameter **LINTG** = SELECTED_INT_KIND(18)
Long integer kind.
- INTEGER, parameter **PTR** = **INTG**
Pointer integer kind.
- INTEGER, parameter **SP** = SELECTED_REAL_KIND(6)
Single precision real kind.
- INTEGER, parameter **DP** = SELECTED_REAL_KIND(15)
Double precision real kind.
- INTEGER, parameter **QP** = SELECTED_REAL_KIND(30)
Quadruple precision real kind.
- INTEGER, parameter **SPC** = KIND((1.0_SP)
- INTEGER, parameter **_SP**
Single precision complex kind.
- INTEGER, parameter **DPC** = KIND((1.0_DP)
- INTEGER, parameter **_DP**
Double precision complex kind.

6.25.1 Detailed Description

This module contains all kind definitions.

6.26 LAPACK Namespace Reference

This module contains the [interface](#) descriptions to the [LAPACK](#) routines.

Classes

- interface [interface](#)

6.26.1 Detailed Description

This module contains the [interface](#) descriptions to the [LAPACK](#) routines.

6.27 LAPLACE_EQUATIONS_ROUTINES Namespace Reference

This module handles all Laplace equations routines.

Functions

- subroutine [`LAPLACE_EQUIV_EQUATIONS_SETFINITEELEMENTCALCULATE`](#) (`EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*`)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.
- subroutine [`LAPLACE_EQUIV_EQUATIONS_SETSETUP`](#) (`EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*`)

Sets up the Laplace equation type of a classical field equations set class.
- subroutine [`LAPLACE_EQUIV_EQUATIONS_SETSUBTYPESET`](#) (`EQUATIONS_SET, EQUATIONS_SET_SUBTYPE, ERR, ERROR,*`)

Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.
- subroutine [`LAPLACE_EQUIV_EQUATIONS_SETSTANDARDSETUP`](#) (`EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*`)

Sets up the standard Laplace equation.
- subroutine [`LAPLACE_EQUIV_PROBLEM_SETUP`](#) (`PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*`)

Sets up the Laplace solution.
- subroutine [`LAPLACE_EQUIV_PROBLEM_SUBTYPESET`](#) (`PROBLEM, PROBLEM_SUBTYPE, ERR, ERROR,*`)

Sets/changes the problem subtype for a Laplace equation type .
- subroutine [`LAPLACE_EQUIV_PROBLEM_STANDARDSETUP`](#) (`PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*`)

Sets up the standard Laplace equations solution.

6.27.1 Detailed Description

This module handles all Laplace equations routines.

6.27.2 Function Documentation

- 6.27.2.1 subroutine [`LAPLACE_EQUIV_EQUATIONS_SETFINITEELEMENTCALCULATE`](#)
`(TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET,`
`INTEGER(INTG),intent(in) ELEMENT_NUMBER, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.

Parameters:

EQUATIONS_SET A pointer to the equations set to perform the finite element calculations on

ELEMENT_NUMBER The element number to calculate

ERR The error code

ERROR The error string

Definition at line 88 of file Laplace_equations_routines.f90.

References KINDS::DP, BASE_ROUTINES::ENTERS(), EQUATIONS_SET_-,
 CONSTANTS::EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE, EQUATIONS_SET_-,
 SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE, BASE_-,
 ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_-
 INTERPOLATE_GAUSS(), FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_-
 CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET(),
 FIELD_ROUTINES::FIELD_VALUES_SET_TYPE, and KINDS::PTR.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_-ELEMENT_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.27.2.2 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets up the Laplace equation type of a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set to setup a Laplace equation on.

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 196 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP().

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP().

Here is the call graph for this function:

Here is the caller graph for this function:

6.27.2.3 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets up the standard Laplace equation.

Parameters:

EQUATIONS_SET A pointer to the equations set to setup

SETUP_TYPE The setup type to perform

ACTION_TYPE The action type to perform

ERR The error code

ERROR The error string

Definition at line 276 of file Laplace_equations_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV_SOLUTION_METHOD`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_ANALYTIC_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_DEPENDENT_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUIPMENTS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FIXED_CONDITIONS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_GEOMETRY_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_MATERIALS_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_SOURCE_TYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE`, `FIELD_ROUTINES::FIELD_CREATE_FINISH()`, `FIELD_ROUTINES::FIELD_DEPENDENT_TYPE`, `FIELD_ROUTINES::FIELD_GENERAL_TYPE`, `FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION`, `FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE`, `FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE`, `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE`, and `MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE`.

Referenced by `LAPLACE_EQUATION_EQUATIONS_SET_SETUP()`, and `LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.27.2.4 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SET_SUBTYPE_SET (TYPE(EQUATIONS_SET_TYPE),pointer EQUATIONS_SET, INTEGER(INTG),intent(in) EQUATIONS_SET_SUBTYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.

Parameters:

EQUATIONS_SET A pointer to the equations set to set the equation subtype for

EQUATIONS_SET_SUBTYPE The equation subtype to set

ERR The error code

ERROR The error string

Definition at line 236 of file Laplace_equations_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GENERALISED_LAPLACE_SUBTYPE**, **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_INITIAL_TYPE**, **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION**, **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_LAPLACE_SUBTYPE**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **LAPLACE_EQUATION_EQUATIONS_SET_STANDARD_SETUP()**.

Referenced by **CLASSICAL_FIELD_ROUTINES::CL()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.27.2.5 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP (TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in) SETUP_TYPE, INTEGER(INTG),intent(in) ACTION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets up the Laplace solution.

Parameters:

PROBLEM A pointer to the solutions set to setup a Laplace equation on.

SETUP_TYPE The setup type

ACTION_TYPE The action type

ERR The error code

ERROR The error string

Definition at line 526 of file Laplace_equations_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()**, **PROBLEM_CONSTANTS::PROBLEM_GENERALISED_LAPLACE_SUBTYPE**, and **PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_SUBTYPE**.

Referenced by **CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.27.2.6 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
PROBLEM_STANDARD_SETUP (TYPE(PROBLEM_TYPE),pointer *PROBLEM*,
INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_-
TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]**

Sets up the standard Laplace equations solution.

Parameters:

PROBLEM A pointer to the problem to setup
SETUP_TYPE The setup type to perform
ACTION_TYPE The action type to perform
ERR The error code
ERROR The error string

Definition at line 606 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-CLASSICAL_FIELD_CLASS, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_-EQUATION_TYPE, EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STANDARD_-LAPLACE_SUBTYPE, BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_-ROUTINES::FIELD_STANDARD_VARIABLE_TYPE, PROBLEM_CONSTANTS::PROBLEM_-SETUP_CONTROL_TYPE, PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_-ACTION, PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION, PROBLEM_-CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE, PROBLEM_CONSTANTS::PROBLEM_-SETUP_SOLUTION_TYPE, PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_-TYPE, PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION, PROBLEM_-CONSTANTS::PROBLEM_STANDARD_LAPLACE_SUBTYPE, KINDS::PTR, SOLVER_-ROUTINES::SOLVER_CREATE_FINISH(), SOLVER_ROUTINES::SOLVER_CREATE_START(), SOLVER_ROUTINES::SOLVER_LIBRARY_SET(), SOLVER_ROUTINES::SOLVER_LINEAR_-TYPE, SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY, SOLVER_ROUTINES::SOLVER_-SPARSE_MATRICES, and SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET().

Referenced by LAPLACE_EQUATION_PROBLEM_SETUP(), and LAPLACE_EQUATION_-PROBLEM_SUBTYPE_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.27.2.7 subroutine LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_-
PROBLEM_SUBTYPE_SET (TYPE(PROBLEM_TYPE),pointer *PROBLEM*,
INTEGER(INTG),intent(in) *PROBLEM_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Sets/changes the problem subtype for a Laplace equation type .

Parameters:

PROBLEM A pointer to the problem to set the problem subtype for
PROBLEM_SUBTYPE The problem subtype to set
ERR The error code
ERROR The error string

Definition at line 566 of file Laplace_equations_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP(),
PROBLEM_CONSTANTS::PROBLEM_GENERALISED_LAPLACE_SUBTYPE, PROBLEM_-
CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE, PROBLEM_CONSTANTS::PROBLEM_-
SETUP_START_ACTION, and PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_-
SUBTYPE.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_-
SET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.28 LISTS Namespace Reference

Implements lists of base types.

Classes

- struct [LIST_PTR_TYPE](#)
Buffer type to allow arrays of pointers to a list.
- struct [LIST_TYPE](#)
Contains information on a list.
- interface [LIST_ITEM_ADD](#)
Adds an item to the end of a list.
- interface [LIST_ITEM_IN_LIST](#)
Determines if an item is in a list and returns the position of the item.
- interface [LIST_DETACH_AND_DESTROY](#)
Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.
- interface [LIST_SEARCH](#)
Searches a list for a given value and returns the position in the list if the value exists.
- interface [LIST_SEARCH_LINEAR](#)
Searches a list using the linear search method.
- interface [LIST_SORT](#)
Sorts a list into ascending order.
- interface [LIST_SORT_BUBBLE](#)
Sorts a list into assending order using the bubble sort method.
- interface [LIST_SORT_HEAP](#)
Sorts a list into assending order using the heap sort method.
- interface [LIST_SORT_SHELL](#)
Sorts a list into either assending or descending order using the shell sort method.

Functions

- subroutine [LIST_CREATE_FINISH](#) (LIST, ERR, ERROR,*)
Finishes the creation of a list created with LIST_CREATE_START.
- subroutine [LIST_CREATE_START](#) (LIST, ERR, ERROR,*)
Starts the creation of a list and returns a pointer to the created list.

- subroutine [LIST_DATA_TYPE_SET](#) (LIST, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type for a list.
- subroutine [LIST_DESTROY](#) (LIST, ERR, ERROR,*)

Destroys a list.
- subroutine [LIST_FINALISE](#) (LIST, ERR, ERROR,*)

Finalises a list and deallocates all memory.
- subroutine [LIST_INITIALISE](#) (LIST, ERR, ERROR,*)

Initialises a list and all its components.
- subroutine [LIST_INITIAL_SIZE_SET](#) (LIST, INITIAL_SIZE, ERR, ERROR,*)

Sets/changes the initial size for a list.
- subroutine [LIST_ITEM_ADD_INTG1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of an integer list.
- subroutine [LIST_ITEM_ADD_SP1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of a single precision real list.
- subroutine [LIST_ITEM_ADD_DP1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of a double precision real list.
- subroutine [LIST_ITEM_IN_LIST_INTG1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given integer LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LIST_ITEM_IN_LIST_SP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given single precision real LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LIST_ITEM_IN_LIST_DP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given double precision real LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LIST_ITEM_DELETE](#) (LIST, LIST_ITEM, ERR, ERROR,*)

Deletes the item given by the LIST_ITEM index from the given list.
- subroutine [LIST_NUMBER_OF_ITEMS_GET](#) (LIST, NUMBER_OF_ITEMS, ERR, ERROR,*)

Gets the current number of items in a list.
- subroutine [LIST_DETACH_AND_DESTROY_INTG](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.
- subroutine [LIST_DETACH_AND_DESTROY_SP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

- subroutine [LIST_DETACH_AND_DESTROY_DP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

- subroutine [LIST_REMOVE_DUPLICATES](#) (LIST, ERR, ERROR,*)

Removes duplicate entries from a list. A side effect of this is that the list is sorted.

- subroutine [LIST_SEARCH_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SEARCH_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a single precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SEARCH_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SEARCH_LINEAR_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SEARCH_LINEAR_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SEARCH_LINEAR_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LIST_SORT_INTG_ARRAY](#) (A, ERR, ERROR,*)

Sorts an integer array list into ascending order.

- subroutine [LIST_SORT_SP_ARRAY](#) (A, ERR, ERROR,*)

Sorts an single precision array list into ascending order.

- subroutine [LIST_SORT_DP_ARRAY](#) (A, ERR, ERROR,*)

Sorts an double precision array list into ascending order.

- subroutine [LIST_SORT_BUBBLE_INTG_ARRAY](#) (A, ERR, ERROR,*)

BUBBLE_SORT_INTG performs a bubble sort on an integer array list.

- subroutine [LIST_SORT_BUBBLE_SP_ARRAY](#) (A, ERR, ERROR,*)

BUBBLE_SORT_SP performs a bubble sort on a single precision array list.

- subroutine [LIST_SORT_BUBBLE_DP_ARRAY](#) (A, ERR, ERROR,*)

BUBBLE_SORT_DP performs a bubble sort on a double precision list.
- subroutine [LIST_SORT_HEAP_INTG_ARRAY](#) (A, ERR, ERROR,*)

Sorts an integer array list into assending order using the heap sort method.
- subroutine [LIST_SORT_HEAP_SP_ARRAY](#) (A, ERR, ERROR,*)

Sorts a real single precision array list into assending order using the heap sort method.
- subroutine [LIST_SORT_HEAP_DP_ARRAY](#) (A, ERR, ERROR,*)

Sorts a real double precision array list into assending order using the heap sort method.
- subroutine [LIST_SORT_SHELL_INTG_ARRAY](#) (A, ERR, ERROR,*)

Sorts an integer array list into either assending or descending order using the shell sort method.
- subroutine [LIST_SORT_SHELL_SP_ARRAY](#) (A, ERR, ERROR,*)

Sorts a real single precision array list into either assending or descending order using the shell sort method.
- subroutine [LIST_SORT_SHELL_DP_ARRAY](#) (A, ERR, ERROR,*)

Sorts a real double precision array list into either assending or descending order using the shell sort method.

Variables

- INTEGER(INTG), parameter [LIST_INTG_TYPE](#) = INTEGER_TYPE

Integer data type for a list.
- INTEGER(INTG), parameter [LIST_SP_TYPE](#) = SINGLE_REAL_TYPE

Single precision real data type for a list.
- INTEGER(INTG), parameter [LIST_DP_TYPE](#) = DOUBLE_REAL_TYPE

Double precision real data type for a list.
- INTEGER(INTG), parameter [LIST_UNSORTED_TYPE](#) = 1

Unsorted list type.
- INTEGER(INTG), parameter [LIST_SORT_ASCENDING_TYPE](#) = 2

Ascending order for sort.
- INTEGER(INTG), parameter [LIST_SORT_DESCENDING_TYPE](#) = 3

Descending order for sort.
- INTEGER(INTG), parameter [LIST_BUBBLE_SORT_METHOD](#) = 1

Bubble sort method.
- INTEGER(INTG), parameter [LIST_SHELL_SORT_METHOD](#) = 2

Shell sort method.
- INTEGER(INTG), parameter [LIST_HEAP_SORT_METHOD](#) = 3

Heap sort method.

6.28.1 Detailed Description

Implements lists of base types.

6.28.2 Function Documentation

6.28.2.1 subroutine LISTS::LIST_CREATE_FINISH (TYPE(LIST_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finishes the creation of a list created with LIST_CREATE_START.

See also:

[LISTS::LIST_CREATE_START](#).

Parameters:

LIST A pointer to the list to finish

ERR The error code

ERROR The error string.

Definition at line 192 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), LIST_DP_TYPE, LIST_INTG_TYPE, and LIST_SP_TYPE.

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MESH_-ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_-ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), and SOLVER_-MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.2 subroutine LISTS::LIST_CREATE_START (TYPE(LIST_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Starts the creation of a list and returns a pointer to the created list.

See also:

[LISTS::LIST_CREATE_FINISH](#).

Parameters:

LIST On exit, pointer to the list to create. Must not be associated on entry.

ERR The error code.

ERROR The error string.

Definition at line 243 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LIST_FINALISE()`, `LIST_HEAP_SORT_METHOD`, `LIST_INITIALISE()`, `LIST_INTG_TYPE`, and `LIST_SORT_ASCENDING_TYPE`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`,
`DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`,
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.3 subroutine `LISTS::LIST_DATA_TYPE_SET` (`TYPE(List_Type)`,pointer *LIST*, `INTEGER(INTG),intent(in) DATA_TYPE`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Sets/changes the data type for a list.

Parameters:

LIST A pointer to the list

DATA_TYPE The data type of the list to set

See also:

[LISTS::DataType,LISTS](#)

ERR The error code

ERROR The error string

Definition at line 279 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LIST_DP_TYPE`, `LIST_INTG_TYPE`, and `LIST_SP_TYPE`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`,
`DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`,
`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.4 subroutine `LISTS::LIST_DESTROY` (`TYPE(List_Type)`,pointer *LIST*, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Destroys a list.

Parameters:

LIST A pointer to the list to destroy

ERR The error code

ERROR The error string

Definition at line 323 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LIST_FINALISE()`.

Referenced by `MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()`, `FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()`, `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()`, `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()`, and `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.28.2.5 subroutine `LISTS::LIST_DETACH_AND_DESTROY_DP` (`TYPE(LIST_TYPE)`,pointer `LIST`, `INTEGER(INTG),intent(out) NUMBER_IN_LIST`,
`REAL(DP),dimension(:),pointer LIST_VALUES, INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`**

Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

Parameters:

`LIST` The pointer to the list

`NUMBER_IN_LIST` On exit, the number in the list that has been detached.

`LIST_VALUES` On exit, a pointer to the detached list. Must not be associated on entry.

`ERR` The error code

`ERROR` The error string

Definition at line 931 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LIST_DP_TYPE`, and `LIST_FINALISE()`.

Here is the call graph for this function:

**6.28.2.6 subroutine `LISTS::LIST_DETACH_AND_DESTROY_INTG` (`TYPE(LIST_TYPE)`,pointer `LIST`, `INTEGER(INTG),intent(out) NUMBER_IN_LIST`,
`INTEGER(INTG),dimension(:),pointer LIST_VALUES, INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`**

Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

Parameters:

`LIST` The pointer to the list

NUMBER_IN_LIST On exit, the number in the list that has been detached.

LIST_VALUES On exit, a pointer to the detached list. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 830 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LIST_FINALISE()`, and `LIST_INTG_TYPE`.

Here is the call graph for this function:

6.28.2.7 subroutine LISTS::LIST_DETACH_AND_DESTROY_SP (TYPE(List_Type),pointer LIST, INTEGER(INTG),intent(out) NUMBER_IN_LIST, REAL(SP),dimension(:),pointer LIST_VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The `LIST_VALUES` pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

Parameters:

LIST The pointer to the list

NUMBER_IN_LIST On exit, the number in the list that has been detached.

LIST_VALUES On exit, a pointer to the detached list. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 881 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `LIST_FINALISE()`, and `LIST_SP_TYPE`.

Here is the call graph for this function:

6.28.2.8 subroutine LISTS::LIST_FINALISE (TYPE(List_Type),pointer LIST, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalises a list and deallocates all memory.

Parameters:

LIST A pointer to the list to finalise

ERR The error code

ERROR The error string

Definition at line 351 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by LIST_CREATE_START(), LIST_DESTROY(), LIST_DETACH_AND_DESTROY_DP(), LIST_DETACH_AND_DESTROY_INTG(), and LIST_DETACH_AND_DESTROY_SP().

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.9 subroutine LISTS::LIST_INITIAL_SIZE_SET (TYPE(List_Type),pointer *LIST*, INTEGER(INTG),intent(in) *INITIAL_SIZE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the initial size for a list.

Parameters:

LIST A pointer to the list

INITIAL_SIZE The initial size of the list to set. Must be greater than zero.

ERR The error code

ERROR The error string

Definition at line 417 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET(), DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE(), FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET(), MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE(), MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE(), and SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.10 subroutine LISTS::LIST_INITIALISE (TYPE(List_Type),pointer *LIST*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialises a list and all its components.

Parameters:

LIST A pointer to the list to initialise

ERR The error code

ERROR The error string

Definition at line 380 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by LIST_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.28.2.11 subroutine LISTS::LIST_ITEM_ADD_DP1 (TYPE(LIST_TYPE),pointer
 $LIST$, REAL(DP),intent(in) $ITEM$, INTEGER(INTG),intent(out) ERR ,
 TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]**

Adds an item to the end of a double precision real list.

Parameters:

$LIST$ A pointer to the list

$ITEM$ The item to add

ERR The error code

$ERROR$ The error string

Definition at line 566 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_DP_TYPE.

Here is the call graph for this function:

**6.28.2.12 subroutine LISTS::LIST_ITEM_ADD_INTG1 (TYPE(LIST_TYPE),pointer
 $LIST$, INTEGER(INTG),intent(in) $ITEM$, INTEGER(INTG),intent(out) ERR ,
 TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]**

Adds an item to the end of an integer list.

Parameters:

$LIST$ A pointer to the list

$ITEM$ The item to add

ERR The error code

$ERROR$ The error string

Definition at line 457 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_INTG_TYPE.

Here is the call graph for this function:

**6.28.2.13 subroutine LISTS::LIST_ITEM_ADD_SP1 (TYPE(LIST_TYPE),pointer
 $LIST$, REAL(SP),intent(in) $ITEM$, INTEGER(INTG),intent(out) ERR ,
 TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]**

Adds an item to the end of a single precision real list.

Parameters:

$LIST$ A pointer to the list

$ITEM$ The item to add

ERR The error code

$ERROR$ The error string

Definition at line 511 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_SP_TYPE.

Here is the call graph for this function:

**6.28.2.14 subroutine LISTS::LIST_ITEM_DELETE (TYPE(LIST_TYPE),pointer *LIST*,
INTEGER(INTG),intent(in) *LIST_ITEM*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Deletes the item given by the *LIST_ITEM* index from the given list.

Parameters:

LIST The pointer to the list

LIST_ITEM The position in the list to delete.

ERR The error code

ERROR The error string

Definition at line 749 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), LIST_DP_TYPE, LIST_INTG_TYPE, and LIST_SP_TYPE.

Here is the call graph for this function:

**6.28.2.15 subroutine LISTS::LIST_ITEM_IN_LIST_DP1 (TYPE(LIST_TYPE),pointer
LIST, REAL(DP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST_ITEM*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Determines if *ITEM* is in the given double precision real *LIST*. If it is *LIST_ITEM* is the index in the list. If not *LIST_ITEM* is 0.

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string

Definition at line 707 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_DP_TYPE.

Here is the call graph for this function:

6.28.2.16 subroutine LISTS::LIST_ITEM_IN_LIST_INTG1 (TYPE(LIST_TYPE),pointer
LIST, INTEGER(INTG),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST_ITEM*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Determines if *ITEM* is in the given integer *LIST*. If it is *LIST_ITEM* is the index in the list. If not *LIST_ITEM* is 0.

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string.

Definition at line 621 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_INTG_TYPE.

Here is the call graph for this function:

6.28.2.17 subroutine LISTS::LIST_ITEM_IN_LIST_SP1 (TYPE(LIST_TYPE),pointer
LIST, REAL(SP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *LIST_ITEM*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Determines if *ITEM* is in the given single precision real *LIST*. If it is *LIST_ITEM* is the index in the list. If not *LIST_ITEM* is 0.

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string

Definition at line 664 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and LIST_SP_TYPE.

Here is the call graph for this function:

6.28.2.18 subroutine LISTS::LIST_NUMBER_OF_ITEMS_GET (TYPE(LIST_TYPE),pointer
LIST, INTEGER(INTG),intent(out) *NUMBER_OF_ITEMS*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Gets the current number of items in a list.

Parameters:

LIST A pointer to the list

NUMBER_OF_ITEMS On exit, the current number of items in the list

ERR The error code

ERROR The error string

Definition at line 796 of file lists.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Referenced by **SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.19 subroutine LISTS::LIST_REMOVE_DUPLICATES (TYPE(LIST_TYPE),pointer LIST, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Removes duplicate entries from a list. A side effect of this is that the list is sorted.

Parameters:

LIST The pointer to the list

ERR The error code

ERROR The error string

Definition at line 981 of file lists.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **LIST_DP_TYPE**, **LIST_INTG_TYPE**, and **LIST_SP_TYPE**.

Referenced by **MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET()**, **DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE()**, **FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET()**, **MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()**, **MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()**, and **SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.28.2.20 subroutine LISTS::LIST_SEARCH_DP_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

Parameters:**A** The list to search**VALUE** The value to search for**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.**ERR** The error code**ERROR** The error string

Definition at line 1157 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.21 subroutine LISTS::LIST_SEARCH_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

Parameters:**A** The list to search**VALUE** The value to search for**POSITION** On exit, the position of value in the list. If value does not exist in the list the returned position is zero.**ERR** The error code**ERROR** The error string

Definition at line 1103 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.22 subroutine LISTS::LIST_SEARCH_LINEAR_DP_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

Parameters:**A** The list to search**VALUE** The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1266 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.23 subroutine LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY
  (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE,
   INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1184 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.24 subroutine LISTS::LIST_SEARCH_LINEAR_SP_ARRAY
  (REAL(SP),dimension(:),intent(in) A, REAL(SP),intent(in) VALUE,
   INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1225 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.25 subroutine LISTS::LIST_SEARCH_SP_ARRAY (REAL(SP),dimension(:),intent(in) A, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Searches a single precision real array list `A` for `VALUE`. If the search is successful `POSITION` contains the index of the position of `VALUE` in the list otherwise `POSITION` is zero.

Parameters:

`A` The list to search

`VALUE` The value to search for

`POSITION` On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

`ERR` The error code

`ERROR` The error string

Definition at line 1130 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.26 subroutine LISTS::LIST_SORT_BUBBLE_DP_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

`BUBBLE_SORT_DP` performs a bubble sort on a double precision list.

Parameters:

`A` The list to sort

`ERR` The error code

`ERROR` The error string

Definition at line 1463 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

```
6.28.2.27 subroutine LISTS::LIST_SORT_BUBBLE_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

`BUBBLE_SORT_INTG` performs a bubble sort on an integer array list.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1382 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.28 subroutine LISTS::LIST_SORT_BUBBLE_SP_ARRAY
`(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

`BUBBLE_SORT_SP` performs a bubble sort on a single precision array list.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1422 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.29 subroutine LISTS::LIST_SORT_DP_ARRAY `(REAL(DP),dimension(:),intent(inout) A,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Sorts an double precision array list into ascending order.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1357 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.30 subroutine LISTS::LIST_SORT_HEAP_DP_ARRAY
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts a real double precision array list into assending order using the heap sort method.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1619 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.28.2.31 subroutine LISTS::LIST_SORT_HEAP_INTG_ARRAY
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts an integer array list into assending order using the heap sort method.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1504 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.28.2.32 subroutine LISTS::LIST_SORT_HEAP_SP_ARRAY
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts a real single precision array list into assending order using the heap sort method.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1561 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.28.2.33 subroutine LISTS::LIST_SORT_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts an integer array list into ascending order.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1307 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.34 subroutine LISTS::LIST_SORT_SHELL_DP_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts a real double precision array list into either assending or descending order using the shell sort method.

Parameters:

- ERR** The error code
- ERROR** The error string

Definition at line 1760 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.35 subroutine LISTS::LIST_SORT_SHELL_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sorts an integer array list into either assending or descending order using the shell sort method.

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1677 of file lists.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.28.2.36 subroutine LISTS::LIST_SORT_SHELL_SP_ARRAY
(REAL(SP),dimension(:),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sorts a real single precision array list into either assending or descending order using the shell sort method.

Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1718 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.28.2.37 subroutine LISTS::LIST_SORT_SP_ARRAY (REAL(SP),dimension(:),intent(inout) *A*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
***) [private]**

Sorts an single precision array list into ascending order.

Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1332 of file lists.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.29 MACHINE_CONSTANTS Namespace Reference

This module contains all machine dependent constants for AIX systems.

Variables

- INTEGER(INTG), parameter `MACHINE_TYPE` = IBM_COMPUTER
- INTEGER(INTG), parameter `MACHINE_OS` = AIX_OS
- INTEGER(INTG), parameter `MACHINE_ENDIAN` = LITTLE_ENDIAN_NUMBER
- INTEGER(INTG), parameter `MACHINE_CHAR_FORMAT` = ASCII_CHARACTER
- INTEGER(INTG), parameter `MACHINE_INT_FORMAT` = TWOS_COMPLEMENT_INTEGER
- INTEGER(INTG), parameter `MACHINE_SP_FORMAT` = SPIEEE_NUMBER
- INTEGER(INTG), parameter `MACHINE_DP_FORMAT` = DPIEEE_NUMBER
- INTEGER(INTG), parameter `INTEGER_SIZE` = 4
- INTEGER(INTG), parameter `SHORT_INTEGER_SIZE` = 2
- INTEGER(INTG), parameter `LONG_INTEGER_SIZE` = 8
- INTEGER(INTG), parameter `SINGLE_REAL_SIZE` = 4
- INTEGER(INTG), parameter `DOUBLE_REAL_SIZE` = 8
- INTEGER(INTG), parameter `CHARACTER_SIZE` = 1
- INTEGER(INTG), parameter `LOGICAL_SIZE` = 4
- INTEGER(INTG), parameter `SINGLE_COMPLEX_SIZE` = 8
- INTEGER(INTG), parameter `DOUBLE_COMPLEX_SIZE` = 16
- CHARACTER(LEN=1), parameter `ERROR_SEPARATOR_CONSTANT` = CHAR(6)

6.29.1 Detailed Description

This module contains all machine dependent constants for AIX systems.

This module contains all machine dependent constants for Win32 systems.

This module contains all machine dependent constants for VMS systems.

This module contains all machine dependent constants for Linux systems.

This module contains all machine dependent constants for IRIX systems.

6.29.2 Variable Documentation

6.29.2.1 INTEGER(INTG),parameter MACHINE_CONSTANTS::CHARACTER_SIZE = 1

Definition at line 66 of file machine_constants_aix.f90.

Referenced by `BINARY_FILE::OPEN_CMISS_BINARY_FILE()`, and `BINARY_FILE::SKIP_CM_BINARY_HEADER()`.

6.29.2.2 INTEGER(INTG),parameter MACHINE_CONSTANTS::DOUBLE_COMPLEX_SIZE = 16

Definition at line 69 of file machine_constants_aix.f90.

Referenced by `BINARY_FILE::OPEN_CMISS_BINARY_FILE()`.

6.29.2.3 INTEGER(INTG),parameter MACHINE_CONSTANTS::DOUBLE_REAL_SIZE = 8

Definition at line 65 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.4 CHARACTER(LEN=1),parameter MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT = CHAR(6)

Definition at line 71 of file machine_constants_aix.f90.

Referenced by CMISS::CMISS_WRITE_ERROR(), and BASE_ROUTINES::ERRORS().

6.29.2.5 INTEGER(INTG),parameter MACHINE_CONSTANTS::INTEGER_SIZE = 4

Definition at line 61 of file machine_constants_aix.f90.

Referenced by F90C::F2CSTRING(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), BINARY_FILE::RESET_BINARY_NUMBER_TAGS(), and BINARY_FILE::SKIP_CM_BINARY_HEADER().

6.29.2.6 INTEGER(INTG),parameter MACHINE_CONSTANTS::LOGICAL_SIZE = 4

Definition at line 67 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.7 INTEGER(INTG),parameter MACHINE_CONSTANTS::LONG_INTEGER_SIZE = 8

Definition at line 63 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.8 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_CHAR_FORMAT = ASCII_CHARACTER

Definition at line 57 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.9 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_DP_FORMAT = DPIEEE_NUMBER

Definition at line 60 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.10 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_ENDIAN = LITTLE_ENDIAN_NUMBER

Definition at line 56 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_BINARY_FILE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), BINARY_FILE::READ_BINARY_FILE_DP(), BINARY_FILE::READ_BINARY_FILE_DP1(), BINARY_FILE::READ_BINARY_FILE_DPC(), BINARY_FILE::READ_BINARY_FILE_DPC1(), BINARY_FILE::READ_BINARY_FILE_INTG(), BINARY_FILE::READ_BINARY_FILE_INTG1(), BINARY_FILE::READ_BINARY_FILE_LINTG(), BINARY_FILE::READ_BINARY_FILE_LINTG1(), BINARY_FILE::READ_BINARY_FILE_LOGICAL(), BINARY_FILE::READ_BINARY_FILE_LOGICAL1(), BINARY_FILE::READ_BINARY_FILE_SINTG(), BINARY_FILE::READ_BINARY_FILE_SINTG1(), BINARY_FILE::READ_BINARY_FILE_SP(), BINARY_FILE::READ_BINARY_FILE_SP1(), BINARY_FILE::READ_BINARY_FILE_SPC(), and BINARY_FILE::READ_BINARY_FILE_SPC1().

6.29.2.11 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_INT_FORMAT = TWOS_COMPLEMENT_INTEGER

Definition at line 58 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.12 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_OS = AIX_OS

Definition at line 55 of file machine_constants_aix.f90.

Referenced by BASE_ROUTINES::BASE_ROUTINES_INITIALISE(), BINARY_FILE::OPEN_CMISS_BINARY_FILE(), and BASE_ROUTINES::WRITE_STR().

6.29.2.13 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_SP_FORMAT = SPIEEE_NUMBER

Definition at line 59 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.14 INTEGER(INTG),parameter MACHINE_CONSTANTS::MACHINE_TYPE = IBM_COMPUTER

Definition at line 54 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.15 INTEGER(INTG),parameter MACHINE_CONSTANTS::SHORT_INTEGER_SIZE = 2

Definition at line 62 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.16 INTEGER(INTG),parameter MACHINE_CONSTANTS::SINGLE_COMPLEX_SIZE = 8

Definition at line 68 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE().

6.29.2.17 INTEGER(INTG),parameter MACHINE_CONSTANTS::SINGLE_REAL_SIZE = 4

Definition at line 64 of file machine_constants_aix.f90.

Referenced by BINARY_FILE::OPEN_CMISS_BINARY_FILE(), BINARY_FILE::RESET_BINARY_-NUMBER_TAGS(), and BINARY_FILE::SKIP_CM_BINARY_HEADER().

6.30 MATHS Namespace Reference

This module contains all mathematics support routines.

Classes

- interface [CROSS_PRODUCT](#)
- interface [D_CROSS_PRODUCT](#)
- interface [DETERMINANT](#)
- interface [EDP](#)
- interface [EIGENVALUE](#)
- interface [EIGENVECTOR](#)
- interface [I0](#)
- interface [I1](#)
- interface [INVERT](#)
- interface [K0](#)
- interface [K1](#)
- interface [L2NORM](#)
- interface [NORMALISE](#)
- interface [SOLVE_SMALL_LINEAR_SYSTEM](#)

Functions

- subroutine [CROSS_PRODUCT_INTG](#) (A, B, C, ERR, ERROR,*)
- subroutine [CROSS_PRODUCT_SP](#) (A, B, C, ERR, ERROR,*)
- subroutine [CROSS_PRODUCT_DP](#) (A, B, C, ERR, ERROR,*)
- subroutine [D_CROSS_PRODUCT_INTG](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [D_CROSS_PRODUCT_SP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [D_CROSS_PRODUCT_DP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- INTEGER(INTG) [DETERMINANT_FULL_INTG](#) (A, ERR, ERROR)
- REAL(SP) [DETERMINANT_FULL_SP](#) (A, ERR, ERROR)
- REAL(DP) [DETERMINANT_FULL_DP](#) (A, ERR, ERROR)
- REAL(DP) [EDP_DP](#) (X)
- REAL(SP) [EDP_SP](#) (X)
- subroutine [EIGENVALUE_FULL_SP](#) (A, EVALUES, ERR, ERROR,*)
- subroutine [EIGENVALUE_FULL_DP](#) (A, EVALUES, ERR, ERROR,*)
- subroutine [EIGENVECTOR_FULL_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)
- subroutine [EIGENVECTOR_FULL_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)
- REAL(DP) [I0_DP](#) (X)
- REAL(SP) [I0_SP](#) (X)
- REAL(DP) [I1_DP](#) (X)
- REAL(SP) [I1_SP](#) (X)
- subroutine [INVERT_FULL_SP](#) (A, B, DET, ERR, ERROR,*)
- subroutine [INVERT_FULL_DP](#) (A, B, DET, ERR, ERROR,*)
- REAL(DP) [K0_DP](#) (X)
- REAL(SP) [K0_SP](#) (X)
- REAL(DP) [K1_DP](#) (X)
- REAL(SP) [K1_SP](#) (X)
- REAL(DP) [KDP_DP](#) (X)

- REAL(SP) [KDP_SP](#) (X)
- REAL(SP) [L2NORM_SP](#) (A)
- REAL(DP) [L2NORM_DP](#) (A)
- REAL(SP) [NORMALISE_SP](#) (A, ERR, ERROR)
- REAL(DP) [NORMALISE_DP](#) (A, ERR, ERROR)
- subroutine [SOLVE_SMALL_LINEAR_SYSTEM_SP](#) (A, x, b, ERR, ERROR,*)
- subroutine [SOLVE_SMALL_LINEAR_SYSTEM_DP](#) (A, x, b, ERR, ERROR,*)

6.30.1 Detailed Description

This module contains all mathematics support routines.

6.30.2 Function Documentation

6.30.2.1 subroutine MATHS::CROSS_PRODUCT_DP (REAL(DP),dimension(:),intent(in) A, REAL(DP),dimension(:),intent(in) B, REAL(DP),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 239 of file maths.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.30.2.2 subroutine MATHS::CROSS_PRODUCT_INTG (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),dimension(:),intent(in) B, INTEGER(INTG),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 151 of file maths.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.30.2.3 subroutine MATHS::CROSS_PRODUCT_SP (REAL(SP),dimension(:),intent(in) A, REAL(SP),dimension(:),intent(in) B, REAL(SP),dimension(:),intent(out) C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 195 of file maths.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.30.2.4 subroutine MATHS::D_CROSS_PRODUCT_DP (INTEGER(INTG),intent(in) N , REAL(DP),dimension(:,),intent(in) A , REAL(DP),dimension(:,),intent(in) B , REAL(DP),dimension(:,),intent(out) C , REAL(DP),dimension(:,;),intent(in) D_A , REAL(DP),dimension(:,;),intent(in) D_B , REAL(DP),dimension(:,;),intent(out) D_C , INTEGER(INTG),intent(out) ERR , TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]

Definition at line 404 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.5 subroutine MATHS::D_CROSS_PRODUCT_INTG (INTEGER(INTG),intent(in) N , INTEGER(INTG),dimension(:,),intent(in) A , INTEGER(INTG),dimension(:,),intent(in) B , INTEGER(INTG),dimension(:,),intent(out) C , INTEGER(INTG),dimension(:,;),intent(in) D_A , INTEGER(INTG),dimension(:,;),intent(in) D_B , INTEGER(INTG),dimension(:,;),intent(out) D_C , INTEGER(INTG),intent(out) ERR , TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]

Definition at line 293 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.6 subroutine MATHS::D_CROSS_PRODUCT_SP (INTEGER(INTG),intent(in) N , REAL(SP),dimension(:,),intent(in) A , REAL(SP),dimension(:,),intent(in) B , REAL(SP),dimension(:,),intent(out) C , REAL(SP),dimension(:,;),intent(in) D_A , REAL(SP),dimension(:,;),intent(in) D_B , REAL(SP),dimension(:,;),intent(out) D_C , INTEGER(INTG),intent(out) ERR , TYPE(VARYING_STRING),intent(out) $ERROR$, *) [private]

Definition at line 348 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.7 REAL(DP) MATHS::DETERMINANT_FULL_DP (REAL(DP),dimension(:,;),intent(in) A , INTEGER(INTG),intent(out) ERR , TYPE(VARYING_STRING),intent(out) $ERROR$) [private]

Definition at line 561 of file maths.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.30.2.8 INTEGER(INTG) MATHS::DETERMINANT_FULL_INTG
(INTEGER(INTG),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Definition at line 469 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.9 REAL(SP) MATHS::DETERMINANT_FULL_SP (REAL(SP),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 515 of file maths.f90.

References KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.10 REAL(DP) MATHS::EDP_DP (REAL(DP),intent(in) *X*) [private]

Definition at line 616 of file maths.f90.

References KINDS::_DP.

6.30.2.11 REAL(SP) MATHS::EDP_SP (REAL(SP),intent(in) *X*) [private]

Definition at line 652 of file maths.f90.

References KINDS::_SP.

**6.30.2.12 subroutine MATHS::EIGENVALUE_FULL_DP (REAL(DP),dimension(:,:),intent(in) *A*,
REAL(DP),dimension(:,intent(out) *EVALUES*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 789 of file maths.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.30.2.13 subroutine MATHS::EIGENVALUE_FULL_SP (REAL(SP),dimension(:,:),intent(in) *A*,
REAL(SP),dimension(:,intent(out) *EVALUES*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Definition at line 697 of file maths.f90.

References KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.14 subroutine MATHS::EIGENVECTOR_FULL_DP (REAL(DP),dimension(:,:),intent(in) A, REAL(DP),intent(in) EVALUE, REAL(DP),dimension(:,intent(out) EVECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 981 of file maths.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.15 subroutine MATHS::EIGENVECTOR_FULL_SP (REAL(SP),dimension(:,:),intent(in) A, REAL(SP),intent(in) EVALUE, REAL(SP),dimension(:,intent(out) EVECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 890 of file maths.f90.

References KINDS::_DP, KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.16 REAL(DP) MATHS::I0_DP (REAL(DP),intent(in) X) [private]

Definition at line 1081 of file maths.f90.

References KINDS::_DP.

6.30.2.17 REAL(SP) MATHS::I0_SP (REAL(SP),intent(in) X) [private]

Definition at line 1116 of file maths.f90.

References KINDS::_SP.

6.30.2.18 REAL(DP) MATHS::I1_DP (REAL(DP),intent(in) X) [private]

Definition at line 1160 of file maths.f90.

References KINDS::_DP.

6.30.2.19 REAL(SP) MATHS::I1_SP (REAL(SP),intent(in) X) [private]

Definition at line 1195 of file maths.f90.

References KINDS::_SP.

6.30.2.20 subroutine MATHS::INVERT_FULL_DP (REAL(DP),dimension(:,:),intent(in) A, REAL(DP),dimension(:, :, intent(out) B, REAL(DP),intent(out) DET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1316 of file maths.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.21 subroutine MATHS::INVERT_FULL_SP (REAL(SP),dimension(:,:),intent(in) A, REAL(SP),dimension(:,:),intent(out) B, REAL(SP),intent(out) DET, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1239 of file maths.f90.

References KINDS::_DP, KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.30.2.22 REAL(DP) MATHS::K0_DP (REAL(DP),intent(in) X) [private]

Definition at line 1402 of file maths.f90.

References KINDS::_DP.

6.30.2.23 REAL(SP) MATHS::K0_SP (REAL(SP),intent(in) X) [private]

Definition at line 1453 of file maths.f90.

References KINDS::_SP.

6.30.2.24 REAL(DP) MATHS::K1_DP (REAL(DP),intent(in) X) [private]

Definition at line 1513 of file maths.f90.

References KINDS::_DP.

6.30.2.25 REAL(SP) MATHS::K1_SP (REAL(SP),intent(in) X) [private]

Definition at line 1565 of file maths.f90.

References KINDS::_SP.

6.30.2.26 REAL(DP) MATHS::KDP_DP (REAL(DP),intent(in) X) [private]

Definition at line 1626 of file maths.f90.

References KINDS::_DP.

6.30.2.27 REAL(SP) MATHS::KDP_SP (REAL(SP),intent(in) X) [private]

Definition at line 1664 of file maths.f90.

References KINDS::_SP.

**6.30.2.28 REAL(DP) MATHS::L2NORM_DP (REAL(DP),dimension(:),intent(in) A)
[private]**

Definition at line 1735 of file maths.f90.

6.30.2.29 REAL(SP) MATHS::L2NORM_SP (REAL(SP),dimension(:),intent(in) A) [private]

Definition at line 1711 of file maths.f90.

**6.30.2.30 REAL(DP) MATHS::NORMALISE_DP (REAL(DP),dimension(:),intent(in) A,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Definition at line 1805 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.30.2.31 REAL(SP) MATHS::NORMALISE_SP (REAL(SP),dimension(:),intent(in) A,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Definition at line 1768 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.30.2.32 subroutine MATHS::SOLVE_SMALL_LINEAR_SYSTEM_DP
(REAL(DP),dimension(:, :, intent(in) A, REAL(DP),dimension(:, intent(out)
x, REAL(DP),dimension(:, intent(in) b, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Definition at line 1900 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.30.2.33 subroutine MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP
(REAL(SP),dimension(:, :, intent(in) A, REAL(SP),dimension(:, intent(out)
x, REAL(SP),dimension(:, intent(in) b, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Definition at line 1851 of file maths.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

6.31 MATRIX_VECTOR Namespace Reference

This module contains all routines dealing with (non-distributed) matrix and vectors types.

Classes

- interface [MATRIX_ALL_VALUES_SET](#)
- interface [MATRIX_DATA_GET](#)
- interface [MATRIX_VALUES_ADD](#)
- interface [MATRIX_VALUES_GET](#)
- interface [MATRIX_VALUES_SET](#)
- interface [VECTOR_ALL_VALUES_SET](#)
- interface [VECTOR_DATA_GET](#)
- interface [VECTOR_VALUES_GET](#)
- interface [VECTOR_VALUES_SET](#)

Functions

- subroutine [MATRIX_ALL_VALUES_SET_INTG](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in an integer matrix to the specified value.
- subroutine [MATRIX_ALL_VALUES_SET_SP](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a single precision matrix to the specified value.
- subroutine [MATRIX_ALL_VALUES_SET_DP](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a double precision matrix to the specified value.
- subroutine [MATRIX_ALL_VALUES_SET_L](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a logical matrix to the specified value.
- subroutine [MATRIX_CREATE_FINISH](#) (MATRIX, ERR, ERROR,*)

Finishes the creation a matrix.
- subroutine [MATRIX_CREATE_START](#) (MATRIX, ERR, ERROR,*)

Starts the creation a matrix.
- subroutine [MATRIX_DATA_GET_INTG](#) (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_DATA_GET_SP](#) (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_DATA_GET_DP](#) (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

- subroutine [MATRIX_DATA_GET_L](#) (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_DATA_TYPE_SET](#) (MATRIX, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type of a matrix.
- subroutine [MATRIX_DESTROY](#) (MATRIX, ERR, ERROR,*)

Destroys a matrix.
- subroutine [MATRIX_DUPLICATE](#) (MATRIX, NEW_MATRIX, ERR, ERROR,*)

Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.
- subroutine [MATRIX_FINALISE](#) (MATRIX, ERR, ERROR,*)

Finalises a matrix and deallocates all memory.
- subroutine [MATRIX_INITIALISE](#) (MATRIX, ERR, ERROR,*)

Initialises a matrix.
- subroutine [MATRIX_MAX_COLUMNS_PER_ROW_GET](#) (MATRIX, MAX_COLUMNS_PER_ROW, ERR, ERROR,*)

Gets the maximum number of columns in each row of a distributed matrix.
- subroutine [MATRIX_NUMBER_NON_ZEROS_SET](#) (MATRIX, NUMBER_NON_ZEROS, ERR, ERROR,*)

Sets/changes the number of non zeros for a matrix.
- subroutine [MATRIX_MAX_SIZE_SET](#) (MATRIX, MAX_M, MAX_N, ERR, ERROR,*)

Sets/changes the maximum size of a matrix.
- subroutine [MATRIX_OUTPUT](#) (ID, MATRIX, ERR, ERROR,*)

Sets/changes the size of a matrix.
- subroutine [MATRIX_SIZE_SET](#) (MATRIX, M, N, ERR, ERROR,*)

Sets/changes the size of a matrix.
- subroutine [MATRIX_STORAGE_LOCATION_FIND](#) (MATRIX, I, J, LOCATION, ERR, ERROR,*)

Returns the storage location in the data array of a matrix that corresponds to location I,J. If the location does not exist the routine returns zero.
- subroutine [MATRIX_STORAGE_LOCATIONS_GET](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)

Gets the storage locations (sparsity pattern) of a matrix.
- subroutine [MATRIX_STORAGE_LOCATIONS_SET](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)

Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.
- subroutine [MATRIX_STORAGE_TYPE_GET](#) (MATRIX, STORAGE_TYPE, ERR, ERROR,*)

Gets the storage type for a matrix.

- subroutine [MATRIX_STORAGE_TYPE_SET](#) (MATRIX, STORAGE_TYPE, ERR, ERROR,*)

Sets/changes the storage type for a matrix.
- subroutine [MATRIX_VALUES_ADD_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to an integer matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to an integer matrix at the location specified by the row and column index i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a single precision real matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a single precision real matrix at the location specified by the row and column index i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_DP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a double precision real matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_DP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a double precision real matrix at the location specified by the row and column index i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_DP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.
- subroutine [MATRIX_VALUES_ADD_L](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a logical matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J)+VALUE$.

Adds values to a logical matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$.

- subroutine [MATRIX_VALUES_ADD_L1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a logical matrix at the location specified by the row and column index i.e., $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$.

- subroutine [MATRIX_VALUES_ADD_L2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e., $MATRIX(I,J)=MATRIX(I,J).OR.VALUE$.

- subroutine [MATRIX_VALUES_GET_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets the values in an integer matrix at the location specified by the row and column indices i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in an integer matrix at the location specified by the row and column index i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a single precision real matrix at the location specified by the row and column indices i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a single precision real matrix at the location specified by the row and column index i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_DP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a double precision real matrix at the location specified by the row and column indices i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_DP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a double precision real matrix at the location specified by the row and column index i.e., $VALUE=MATRIX(I,J)$.

- subroutine [MATRIX_VALUES_GET_DP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine [MATRIX_VALUES_GET_L](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine [MATRIX_VALUES_GET_L1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a logical matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).
- subroutine [MATRIX_VALUES_GET_L2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine [MATRIX_VALUES_SET_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.
- subroutine [MATRIX_VALUES_SET_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.
- subroutine [MATRIX_VALUES_SET_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.
- subroutine [MATRIX_VALUES_SET_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.
- subroutine [MATRIX_VALUES_SET_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets the value in a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.
- subroutine [MATRIX_VALUES_SET_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.

- subroutine [MATRIX_VALUES_SET_DP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [MATRIX_VALUES_SET_DPI](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in a double precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [MATRIX_VALUES_SET_DP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [MATRIX_VALUES_SET_L](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [MATRIX_VALUES_SET_L1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in a logical matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [MATRIX_VALUES_SET_L2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.
- subroutine [VECTOR_ALL_VALUES_SET_INTG](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in an integer vector to the specified value.
- subroutine [VECTOR_ALL_VALUES_SET_SP](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a single precision vector to the specified value.
- subroutine [VECTOR_ALL_VALUES_SET_DP](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a double precision vector to the specified value.
- subroutine [VECTOR_ALL_VALUES_SET_L](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a logical vector to the specified value.
- subroutine [VECTOR_CREATE_FINISH](#) (VECTOR, ERR, ERROR,*)

Finishes the creation of a vector.
- subroutine [VECTOR_CREATE_START](#) (VECTOR, ERR, ERROR,*)

Starts the creation a vector.
- subroutine [VECTOR_DATA_GET_INTG](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

- subroutine [VECTOR_DATA_GET_SP](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [VECTOR_DATA_GET_DP](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [VECTOR_DATA_GET_L](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [VECTOR_DATA_TYPE_SET](#) (VECTOR, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type of a vector.
- subroutine [VECTOR_DESTROY](#) (VECTOR, ERR, ERROR,*)

Destroys a vector.
- subroutine [VECTOR_DUPLICATE](#) (VECTOR, NEW_VECTOR, ERR, ERROR,*)

Duplicates a vector structure and returns a pointer to the new vector in NEW_VECTOR.
- subroutine [VECTOR_FINALISE](#) (VECTOR, ERR, ERROR,*)

Finalises a vector and deallocates all memory.
- subroutine [VECTOR_INITIALISE](#) (VECTOR, ERR, ERROR,*)

Initialises a vector.
- subroutine [VECTOR_SIZE_SET](#) (VECTOR, N, ERR, ERROR,*)

Sets/changes the size of a vector.
- subroutine [VECTOR_VALUES_GET_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in an integer vector at the indices specified.
- subroutine [VECTOR_VALUES_GET_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in an integer vector at the location specified by the index.
- subroutine [VECTOR_VALUES_GET_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a single precision real vector at the indices specified.
- subroutine [VECTOR_VALUES_GET_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a single precision vector at the location specified by the index.
- subroutine [VECTOR_VALUES_GET_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a double precision real vector at the indices specified.
- subroutine [VECTOR_VALUES_GET_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a double precision vector at the location specified by the index.

- subroutine [VECTOR_VALUES_GET_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a logical real vector at the indices specified.
- subroutine [VECTOR_VALUES_GET_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a logical vector at the location specified by the index.
- subroutine [VECTOR_VALUES_SET_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in an integer vector at the specified indices.
- subroutine [VECTOR_VALUES_SET_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in an integer vector at the specified index.
- subroutine [VECTOR_VALUES_SET_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a single precision vector at the specified indices.
- subroutine [VECTOR_VALUES_SET_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a single precision vector at the specified index.
- subroutine [VECTOR_VALUES_SET_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a double precision vector at the specified indices.
- subroutine [VECTOR_VALUES_SET_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a double precision vector at the specified index.
- subroutine [VECTOR_VALUES_SET_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a logical vector at the specified indices.
- subroutine [VECTOR_VALUES_SET_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a logical vector at the specified index.

Variables

- INTEGER(INTG), parameter [MATRIX_VECTOR_INTG_TYPE](#) = INTEGER_TYPE

Integer matrix-vector data type.
- INTEGER(INTG), parameter [MATRIX_VECTOR_SP_TYPE](#) = SINGLE_REAL_TYPE

Single precision real matrix-vector data type.
- INTEGER(INTG), parameter [MATRIX_VECTOR_DP_TYPE](#) = DOUBLE_REAL_TYPE

Double precision real matrix-vector data type.
- INTEGER(INTG), parameter [MATRIX_VECTOR_L_TYPE](#) = LOGICAL_TYPE

Logical matrix-vector data type.
- INTEGER(INTG), parameter [MATRIX_BLOCK_STORAGE_TYPE](#) = 0

Matrix block storage type.
- INTEGER(INTG), parameter [MATRIX_DIAGONAL_STORAGE_TYPE](#) = 1

Matrix diagonal storage type.

- INTEGER(INTG), parameter **MATRIX_COLUMN_MAJOR_STORAGE_TYPE** = 2
Matrix column major storage type.
- INTEGER(INTG), parameter **MATRIX_ROW_MAJOR_STORAGE_TYPE** = 3
Matrix row major storage type.
- INTEGER(INTG), parameter **MATRIX_COMPRESSED_ROW_STORAGE_TYPE** = 4
Matrix compressed row storage type.
- INTEGER(INTG), parameter **MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE** = 5
Matrix compressed column storage type.
- INTEGER(INTG), parameter **MATRIX_ROW_COLUMN_STORAGE_TYPE** = 6
Matrix row-column storage type.
- INTEGER(INTG), save **MATRIX_VECTOR_ID** = 1

6.31.1 Detailed Description

This module contains all routines dealing with (non-distributed) matrix and vectors types.

6.31.2 Function Documentation

6.31.2.1 subroutine **MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP** `(TYPE(MATRIX_TYPE),pointer MATRIX, REAL(DP),intent(in) VALUE,` `INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in a double precision matrix to the specified value.

Parameters:

MATRIX A pointer to the matrix

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 373 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.2 subroutine **MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_INTG** `(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) VALUE,` `INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in an integer matrix to the specified value.

Parameters:

MATRIX A pointer to the matrix
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 293 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.3 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L
`(TYPE(MATRIX_TYPE),pointer MATRIX, LOGICAL,intent(in) VALUE,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in a logical matrix to the specified value.

Parameters:

MATRIX A pointer to the matrix
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 413 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.4 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_SP
`(TYPE(MATRIX_TYPE),pointer MATRIX, REAL(SP),intent(in) VALUE,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in a single precision matrix to the specified value.

Parameters:

MATRIX A pointer to the matrix
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 333 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.5 subroutine MATRIX_VECTOR::MATRIX_CREATE_FINISH
(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Finishes the creation a matrix.

Parameters:

MATRIX A pointer to the matrix to finish

ERR The error code

ERROR The error string

Definition at line 453 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_-STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_-COMPRESSED_ROW_STORAGE_TYPE, MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_-ROW_COLUMN_STORAGE_TYPE, MATRIX_ROW_MAJOR_STORAGE_TYPE, MATRIX_-VECTOR_DP_TYPE, MATRIX_VECTOR_ID, MATRIX_VECTOR_INTG_TYPE, MATRIX_-VECTOR_L_TYPE, and MATRIX_VECTOR_SP_TYPE.

Referenced by MATRIX_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.6 subroutine MATRIX_VECTOR::MATRIX_CREATE_START
(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Starts the creation a matrix.

Parameters:

MATRIX A pointer to the matrix

ERR The error code

ERROR The error string

Definition at line 588 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_FINALISE(), MATRIX_-INITIALISE(), and MATRIX_VECTOR_DP_TYPE.

Referenced by MATRIX_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.7 subroutine MATRIX_VECTOR::MATRIX_DATA_GET_DP **(TYPE(MATRIX_-TYPE),pointer MATRIX, REAL(DP),dimension(:),pointer DATA,**
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be

deallocated.

Parameters:

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 712 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.8 subroutine MATRIX_VECTOR::MATRIX_DATA_GET_INTG (TYPE(MATRIX_-TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

Parameters:

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 622 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.9 subroutine MATRIX_VECTOR::MATRIX_DATA_GET_L (TYPE(MATRIX_-TYPE),pointer MATRIX, LOGICAL,dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a [MATRIX_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

Parameters:

- MATRIX** A pointer to the matrix
- DATA** On return a pointer to the matrix data
- ERR** The error code
- ERROR** The error string

Definition at line 757 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.10 subroutine MATRIX_VECTOR::MATRIX_DATA_GET_SP (TYPE(MATRIX_TYPE),pointer MATRIX, REAL(SP),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but aMATRIX_VALUES_SET call must be used to change any values. The pointer should not be deallocated.

Parameters:

MATRIX A pointer to the matrix
DATA On return a pointer to the matrix data
ERR The error code
ERROR The error string

Definition at line 667 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.11 subroutine MATRIX_VECTOR::MATRIX_DATA_TYPE_SET (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) DATA_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the data type of a matrix.

Parameters:

MATRIX A pointer to the matrix
DATA_TYPE The data type to set for the matrix.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

ERR The error code

ERROR The error string

Definition at line 802 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_VECTOR_DP_TYPE, MATRIX_VECTOR_INTG_TYPE, MATRIX_VECTOR_L_TYPE, and MATRIX_VECTOR_SP_TYPE.

Referenced by MATRIX_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.12 subroutine MATRIX_VECTOR::MATRIX_DESTROY (TYPE(MATRIX_-
TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Destroys a matrix.

Parameters:

MATRIX A pointer to the matrix to destroy

ERR The error code

ERROR The error string

Definition at line 848 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_FINALISE().

Here is the call graph for this function:

**6.31.2.13 subroutine MATRIX_VECTOR::MATRIX_DUPLICATE (TYPE(MATRIX_-
TYPE),pointer *MATRIX*, TYPE(MATRIX_TYPE),pointer *NEW_MATRIX*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)**

Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.

Parameters:

MATRIX A pointer to the matrix to duplicate

NEW_MATRIX On return a pointer to a new duplicated matrix

ERR The error code

ERROR The error string

Definition at line 875 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_-STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_-COMPRESSED_ROW_STORAGE_TYPE, MATRIX_CREATE_FINISH(), MATRIX_CREATE_-START(), MATRIX_DATA_TYPE_SET(), MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_-FINALISE(), MATRIX_MAX_SIZE_SET(), MATRIX_NUMBER_NON_ZEROS_SET(), MATRIX_-ROW_COLUMN_STORAGE_TYPE, MATRIX_ROW_MAJOR_STORAGE_TYPE, MATRIX_SIZE_-SET(), MATRIX_STORAGE_LOCATIONS_SET(), and MATRIX_STORAGE_TYPE_SET().

Here is the call graph for this function:

**6.31.2.14 subroutine MATRIX_VECTOR::MATRIX_FINALISE (TYPE(MATRIX_-
TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Finalises a matrix and deallocates all memory.

Parameters:

MATRIX A pointer to the matrix to finalise

ERR The error code

ERROR The error string

Definition at line 926 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_CREATE_START()`, `MATRIX_DESTROY()`, and `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.15 subroutine `MATRIX_VECTOR::MATRIX_INITIALISE (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Initialises a matrix.

Parameters:

MATRIX A pointer to the matrix

ERR The error code

ERROR The error string

Definition at line 958 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.16 subroutine `MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) MAX_COLUMNS_PER_ROW, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the maximum number of columns in each row of a distributed matrix.

Parameters:

MATRIX A pointer to the matrix

MAX_COLUMNS_PER_ROW On return, the maximum number of columns in each row

ERR The error code

ERROR The error string

Definition at line 997 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.31.2.17 subroutine MATRIX_VECTOR::MATRIX_MAX_SIZE_SET
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) MAX_M,
 INTEGER(INTG),intent(in) MAX_N, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the maximum size of a matrix.

Parameters:

MATRIX A pointer to the matrix
MAX_M The maximum number of rows to set
MAX_N The maximum number of columns to set
ERR The error code
ERROR The error string

Definition at line 1084 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.18 subroutine MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in)
NUMBER_NON_ZEROS, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the number of non zeros for a matrix.

Parameters:

MATRIX A pointer to the matrix
NUMBER_NON_ZEROS The number of non zeros in the matrix to set
ERR The error code
ERROR The error string

Definition at line 1030 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_BLOCK_STORAGE_TYPE`, `MATRIX_COLUMN_MAJOR_STORAGE_TYPE`, `MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE`, `MATRIX_COMPRESSED_ROW_STORAGE_TYPE`, `MATRIX_DIAGONAL_STORAGE_TYPE`, `MATRIX_ROW_COLUMN_STORAGE_TYPE`, and `MATRIX_ROW_MAJOR_STORAGE_TYPE`.

Referenced by `MATRIX_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.19 subroutine MATRIX_VECTOR::MATRIX_OUTPUT (INTEGER(INTG),intent(in) *ID*, TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the size of a matrix.

Parameters:

ID The ID to output to

MATRIX A pointer to the matrix

ERR The error code

ERROR The error string

Definition at line 1144 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_COMPRESSED_ROW_STORAGE_TYPE, MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_ROW_COLUMN_STORAGE_TYPE, MATRIX_ROW_MAJOR_STORAGE_TYPE, MATRIX_VECTOR_DP_TYPE, MATRIX_VECTOR_INTG_TYPE, MATRIX_VECTOR_L_TYPE, MATRIX_VECTOR_SP_TYPE, and INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES.

Here is the call graph for this function:

6.31.2.20 subroutine MATRIX_VECTOR::MATRIX_SIZE_SET (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *M*, INTEGER(INTG),intent(in) *N*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the size of a matrix.

Parameters:

MATRIX A pointer to the matrix

M The number of rows to set

N The number of columns to set

ERR The error code

ERROR The error string

Definition at line 1265 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by MATRIX_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.21 subroutine MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND
 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *I*,
 INTEGER(INTG),intent(in) *J*, INTEGER(INTG),intent(out) *LOCATION*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Returns the storage location in the data array of a matrix that correponds to location I,J. If the location does not exist the routine returns zero.

Parameters:

MATRIX A pointer to the matrix

I The row number of the location to find

J The column number of the location to find

LOCATION On return the location of the specified row and column in the matrix data. If the row and column does not exist in the matrix then zero is returned.

ERR The error code

ERROR The error string

Definition at line 1313 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_COMPRESSED_ROW_STORAGE_TYPE, MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_ROW_COLUMN_STORAGE_TYPE, and MATRIX_ROW_MAJOR_STORAGE_TYPE.

Referenced by MATRIX_VALUES_ADD_DP(), MATRIX_VALUES_ADD_DP1(), MATRIX_VALUES_ADD_DP2(), MATRIX_VALUES_ADD_INTG(), MATRIX_VALUES_ADD_INTG1(), MATRIX_VALUES_ADD_INTG2(), MATRIX_VALUES_ADD_L(), MATRIX_VALUES_ADD_L1(), MATRIX_VALUES_ADD_L2(), MATRIX_VALUES_ADD_SP(), MATRIX_VALUES_ADD_SP1(), MATRIX_VALUES_ADD_SP2(), MATRIX_VALUES_GET_DP(), MATRIX_VALUES_GET_DP1(), MATRIX_VALUES_GET_DP2(), MATRIX_VALUES_GET_INTG(), MATRIX_VALUES_GET_INTG1(), MATRIX_VALUES_GET_INTG2(), MATRIX_VALUES_GET_L(), MATRIX_VALUES_GET_L1(), MATRIX_VALUES_GET_L2(), MATRIX_VALUES_GET_SP(), MATRIX_VALUES_GET_SP1(), MATRIX_VALUES_GET_SP2(), MATRIX_VALUES_SET_DP(), MATRIX_VALUES_SET_DP1(), MATRIX_VALUES_SET_DP2(), MATRIX_VALUES_SET_INTG(), MATRIX_VALUES_SET_INTG1(), MATRIX_VALUES_SET_INTG2(), MATRIX_VALUES_SET_L(), MATRIX_VALUES_SET_L1(), MATRIX_VALUES_SET_L2(), MATRIX_VALUES_SET_SP(), MATRIX_VALUES_SET_SP1(), and MATRIX_VALUES_SET_SP2().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.22 subroutine MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET
 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),pointer
ROW_INDICES, INTEGER(INTG),dimension(:),pointer *COLUMN_INDICES*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Gets the storage locations (sparsity pattern) of a matrix.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). On return, the row index values for the matrix.

COLUMN_INDICES COLUMN_INDICES(i). On return, the column index values for the matrix.

ERR The error code

ERROR The error string

Definition at line 1433 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_COMPRESSED_ROW_STORAGE_TYPE, MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_ROW_COLUMN_STORAGE_TYPE, and MATRIX_ROW_MAJOR_STORAGE_TYPE.

Here is the call graph for this function:

6.31.2.23 subroutine MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index values for the sparsity pattern.

COLUMN_INDICES COLUMN_INDICES(i). The column index values for the sparsity pattern.

ERR The error code

ERROR The error string

Definition at line 1489 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_BLOCK_STORAGE_TYPE, MATRIX_COLUMN_MAJOR_STORAGE_TYPE, MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE, MATRIX_COMPRESSED_ROW_STORAGE_TYPE, MATRIX_DIAGONAL_STORAGE_TYPE, MATRIX_ROW_COLUMN_STORAGE_TYPE, and MATRIX_ROW_MAJOR_STORAGE_TYPE.

Referenced by MATRIX_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.24 subroutine MATRIX_VECTOR::MATRIX_STORAGE_TYPE_GET
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(out) STORAGE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the storage type for a matrix.

Parameters:

MATRIX A pointer to the matrix

STORAGE_TYPE On return, the storage type of the matrix.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

ERR The error code

ERROR The error string

Definition at line 1709 of file matrix_vector.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.31.2.25 subroutine MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) STORAGE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the storage type for a matrix.

Parameters:

MATRIX A pointer to the matrix

STORAGE_TYPE The storage type to set.

See also:

[MATRIX_VECTOR::StorageTypes](#), [MATRIX_VECTOR](#)

ERR The error code

ERROR The error string

Definition at line 1742 of file matrix_vector.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::EXITS\(\)](#), [MATRIX_BLOCK_STORAGE_TYPE](#), [MATRIX_COLUMN_MAJOR_STORAGE_TYPE](#), [MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE](#), [MATRIX_COMPRESSED_ROW_STORAGE_TYPE](#), [MATRIX_DIAGONAL_STORAGE_TYPE](#), [MATRIX_ROW_COLUMN_STORAGE_TYPE](#), and [MATRIX_ROW_MAJOR_STORAGE_TYPE](#).

Referenced by [MATRIX_DUPLICATE\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.26 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_DP (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES, REAL(DP),dimension(:,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Adds values to a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to add

COLUMN_INDICES COLUMN_INDICES(i). The column index for the i'th value to add

VALUES VALUES(i). The value of the i'th value to add

ERR The error code

ERROR The error string

Definition at line 2166 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.27 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_DP1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
 INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(DP),intent(in) VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Adds a value to a double precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index for the value to add

COLUMN_INDEX The column index for the value to add

VALUE The value to add

ERR The error code

ERROR The error string

Definition at line 2232 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.28 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,
 REAL(DP),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to add

COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to add

VALUES VALUES(i,j). The value of the ij'th value to add

ERR The error code

ERROR The error string

Definition at line 2282 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_-TYPE.

Here is the call graph for this function:

6.31.2.29 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to add

COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to add

VALUES VALUES(i). The value of the i'th value to add

ERR The error code

ERROR The error string

Definition at line 1794 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_INTG_-TYPE.

Here is the call graph for this function:

6.31.2.30 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a value to an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index for the value to add
COLUMN_INDEX The column index for the value to add
VALUE The value to add
ERR The error code
ERROR The error string

Definition at line 1860 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_INTG_-TYPE.

Here is the call graph for this function:

6.31.2.31 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)`
`ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,`
`INTEGER(INTG),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to add
COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to add
VALUES VALUES(i,j). The value of the ij'th value to add
ERR The error code
ERROR The error string

Definition at line 1910 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_INTG_-TYPE.

Here is the call graph for this function:

6.31.2.32 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_L
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)`
`ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,`
`LOGICAL,dimension(:,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds values to a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)\text{.OR.} \text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to add

COLUMN_INDICES COLUMN_INDICES(i). The column index for the i'th value to add

VALUES VALUES(i). The value of the i'th value to add

ERR The error code

ERROR The error string

Definition at line 2352 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.33 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_L1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
 INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(in) VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Adds a value to a logical matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J).OR.VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index for the value to add

COLUMN_INDEX The column index for the value to add

VALUE The value to add

ERR The error code

ERROR The error string

Definition at line 2418 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.34 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_L2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,
 LOGICAL,dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J).OR.VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to add

COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to add

VALUES VALUES(i,j). The value of the ij'th value to add

ERR The error code

ERROR The error string

Definition at line 2468 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.35 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_SP
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)`
`ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,`
`REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Adds values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to add

COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to add

VALUES VALUES(i). The value of the i'th value to add

ERR The error code

ERROR The error string

Definition at line 1980 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_SP_-TYPE.

Here is the call graph for this function:

6.31.2.36 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`
`INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(in) VALUE,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Adds a value to a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index for the value to add

COLUMN_INDEX The column index for the value to add

VALUE The value to add

ERR The error code

ERROR The error string

Definition at line 2046 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE()`.

Here is the call graph for this function:

```
6.31.2.37 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2
  (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
   ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
   REAL(SP),dimension(:, :),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the ij 'th value to add

COLUMN_INDICES `COLUMN_INIDICES(j)`. The column index for the ij 'th value to add

VALUES `VALUES(i,j)`. The value of the ij 'th value to add

ERR The error code

ERROR The error string

Definition at line 2096 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE()`.

Here is the call graph for this function:

```
6.31.2.38 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_DP
  (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
   ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
   REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Gets values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{VALUE}=\text{MATRIX}(I,J)$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the i 'th value to get

COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to get
VALUES VALUES(i). On return the value of the i'th value to get
ERR The error code
ERROR The error string

Definition at line 2898 of file matrix_vector.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.39 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_DP1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
 INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(DP),intent(out) VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Gets a value in a double precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index of the value to get
COLUMN_INDEX The column index of the value to get
VALUE On return the value in the matrix at the specified row and column
ERR The error code
ERROR The error string

Definition at line 2962 of file matrix_vector.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.40 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_DP2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,
 REAL(DP),dimension(:, :,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to get

COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to get
VALUES VALUES(i,j). On return the value of the ij'th value to get
ERR The error code
ERROR The error string

Definition at line 3010 of file matrix_vector.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

```
6.31.2.41 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_INTG
  (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in)
   ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
   INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out)
   ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
```

Gets the values in an integer matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the i'th value to get
COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to get
VALUES VALUES(i). On return the value of the i'th value to get
ERR The error code
ERROR The error string

Definition at line 2538 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

```
6.31.2.42 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1
  (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
   INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(out) VALUE,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   *)
```

Gets a value in an integer matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index of the value to get

COLUMN_INDEX The column index of the value to get

VALUE On return the value in the matrix at the specified row and column

ERR The error code

ERROR The error string

Definition at line 2602 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

6.31.2.43 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:, :,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., `VALUE=MATRIX(I,J)`.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the ij'th value to get

COLUMN_INDICES `COLUMN_INIDICES(j)`. The column index for the ij'th value to get

VALUES `VALUES(i,j)`. On return the value of the ij'th value to get

ERR The error code

ERROR The error string

Definition at line 2650 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

6.31.2.44 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_L
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets values in a logical matrix at the location specified by the row and column indices i.e., `VALUE=MATRIX(I,J)`.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the i'th value to get

COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to get
VALUES VALUES(i). On return the value of the i'th value to get
ERR The error code
ERROR The error string

Definition at line 3078 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.45 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_L1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,
 INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(out) VALUE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Gets a value in a logical matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index of the value to get
COLUMN_INDEX The column index of the value to get
VALUE On return the value in the matrix at the specified row and column
ERR The error code
ERROR The error string

Definition at line 3142 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.46 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_L2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)
 ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,
 LOGICAL,dimension(:, :,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to get
COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to get

VALUES VALUES(i,j). On return the value of the ij'th value to get

ERR The error code

ERROR The error string

Definition at line 3190 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.47 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_SP
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets values in a single precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to get

COLUMN_INDICES COLUMN_INDICES(i). The column index for the i'th value to get

VALUES VALUES(i). On return the value of the i'th value to get

ERR The error code

ERROR The error string

Definition at line 2718 of file matrix_vector.f90.

References KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MATRIX_STORAGE_LOCATION_FIND(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.48 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET_SP1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a single precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to get

COLUMN_INDEX The column index of the value to get

VALUE On return the value in the matrix at the specified row and column

ERR The error code

ERROR The error string

Definition at line 2782 of file matrix_vector.f90.

References `KINDS::_SP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.49 subroutine `MATRIX_VECTOR::MATRIX_VALUES_GET_SP2`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:, :,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{VALUE} = \text{MATRIX}(I, J)$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the i^{th} value to get

COLUMN_INDICES `COLUMN_INDICES(j)`. The column index for the i^{th} value to get

VALUES `VALUES(i, j)`. On return the value of the i^{th} value to get

ERR The error code

ERROR The error string

Definition at line 2830 of file matrix_vector.f90.

References `KINDS::_SP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.50 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_DP`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(DP),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I, J) = \text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix to set.

ROW_INDICES `ROW_INDICES(i)`. The row index of the i^{th} value to set

COLUMN_INDICES `COLUMN_INDICES(i)`. The column index of the i^{th} value to set

VALUES `VALUES(i)`. The value of the i^{th} value to set

ERR The error code

ERROR The error string

Definition at line 3630 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.51 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_DP1`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`
`INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(DP),intent(in) VALUE,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Sets a value in a double precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3696 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.52 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_DP2`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES,`
`INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,`
`REAL(DP),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix to set.

ROW_INDICES `ROW_INDICES(i)`. The row index of the ij'th value to set

COLUMN_INDICES `COLUMN_INDICES(j)`. The column index of the ij'th value to set

VALUES `VALUES(i,j)`. The value of the ij'th value to set

ERR The error code

ERROR The error string

Definition at line 3746 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.53 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in an integer matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the i'th value to set

COLUMN_INDICES `COLUMN_INIDICES(i)`. The column index for the i'th value to set

VALUES `VALUES(i)`. The value of the i'th value to set

ERR The error code

ERROR The error string

Definition at line 3258 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE`.

Here is the call graph for this function:

6.31.2.54 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets a value in an integer matrix at the location specified by the row and column index i.e., `MATRIX(I,J)=VALUE`.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3324 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

6.31.2.55 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:, :, :),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES `ROW_INDICES(i)`. The row index for the i,j 'th value to set

COLUMN_INDICES `COLUMN_INDICES(j)`. The column index for the i,j 'th value to set

VALUES `VALUES(i,j)`. The value of the i,j 'th value to set

ERR The error code

ERROR The error string

Definition at line 3374 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_INTG_TYPE()`.

Here is the call graph for this function:

6.31.2.56 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_L`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES `ROW_INDICES(i)`. The row index of the i 'th value to set

COLUMN_INDICES `COLUMN_INDICES(i)`. The column index of the i 'th value to set

VALUES `VALUES(i)`. The value of the i 'th value to set

ERR The error code

ERROR The error string

Definition at line 3816 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

6.31.2.57 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET_L1
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX,`
`INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(in) VALUE,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Sets a value in a logical matrix at the location specified by the row and column index i.e., `MATRIX(I,J)=VALUE`.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3882 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

6.31.2.58 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET_L2
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in)`
`ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES,`
`LOGICAL,dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e., `MATRIX(I,J)=VALUE`.

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES `ROW_INDICES(i)`. The row index of the ij'th value to set

COLUMN_INDICES `COLUMN_INDICES(j)`. The column index of the ij'th value to set

VALUES `VALUES(i,j)`. The value of the ij'th value to set

ERR The error code

ERROR The error string

Definition at line 3932 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

6.31.2.59 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_SP`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES *ROW_INDICES*(i). The row index of the i'th value to set

COLUMN_INDICES *COLUMN_INDICES*(i). The column index of the i'th value to set

VALUES *VALUES*(i). The value of the i'th value to set

ERR The error code

ERROR The error string

Definition at line 3444 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.60 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_SP1`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the value in a single precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3510 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.61 subroutine `MATRIX_VECTOR::MATRIX_VALUES_SET_SP2`
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, REAL(SP),dimension(:, :,),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES *ROW_INDICES*(i). The row index of the ij'th value to set

COLUMN_INDICES *COLUMN_INDICES*(j). The column index of the ij'th value to set

VALUES *VALUES*(i,j). The value of the ij'th value to set

ERR The error code

ERROR The error string

Definition at line 3560 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_STORAGE_LOCATION_FIND()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.62 subroutine `MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP`
`(TYPE(VECTOR_TYPE),pointer VECTOR, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets all values in a double precision vector to the specified value.

Parameters:

VECTOR A pointer to the vector to set

VALUE The value to set the vector to

ERR The error code

ERROR The error string

Definition at line 4082 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.63 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG
 (TYPE(VECTOR_TYPE),pointer *VECTOR*, INTEGER(INTG),intent(in) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Sets all values in an integer vector to the specified value.

Parameters:

VECTOR A pointer to the vector to set
VALUE The value to set the vector to
ERR The error code
ERROR The error string

Definition at line 4002 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.64 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L
 (TYPE(VECTOR_TYPE),pointer *VECTOR*, LOGICAL,intent(in) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Sets all values in a logical vector to the specified value.

Parameters:

VECTOR A pointer to the vector to set
VALUE The value to set the vector to
ERR The error code
ERROR The error string

Definition at line 4122 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.65 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP
 (TYPE(VECTOR_TYPE),pointer *VECTOR*, REAL(SP),intent(in) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Sets all values in a single precision vector to the specified value.

Parameters:

VECTOR A pointer to the vector to set
VALUE The value to set the vector to

ERR The error code

ERROR The error string

Definition at line 4042 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.66 subroutine MATRIX_VECTOR::VECTOR_CREATE_FINISH
 (TYPE(VECTOR_TYPE),pointer **VECTOR**, INTEGER(INTG),intent(out) **ERR**,
 TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Finishes the creation of a vector.

Parameters:

VECTOR A pointer to the vector

ERR The error code

ERROR The error string

Definition at line 4162 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), MATRIX_VECTOR_DP_TYPE, MATRIX_VECTOR_ID, MATRIX_VECTOR_-
INTG_TYPE, MATRIX_VECTOR_L_TYPE, and MATRIX_VECTOR_SP_TYPE.

Referenced by VECTOR_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.67 subroutine MATRIX_VECTOR::VECTOR_CREATE_START
 (TYPE(VECTOR_TYPE),pointer **VECTOR**, INTEGER(INTG),intent(out) **ERR**,
 TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Starts the creation a vector.

Parameters:

VECTOR A pointer to the vector

ERR The error code

ERROR The error string

Definition at line 4216 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), MATRIX_VECTOR_DP_TYPE, VECTOR_FINALISE(), and VECTOR_-
INITIALISE().

Referenced by VECTOR_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.68 subroutine MATRIX_VECTOR::VECTOR_DATA_GET_DP
`(TYPE(VECTOR_TYPE),pointer VECTOR, REAL(DP),dimension(:),pointer DATA,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4339 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.69 subroutine MATRIX_VECTOR::VECTOR_DATA_GET_INTG
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4249 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, and `MATRIX_VECTOR_INTG_TYPE`.

Here is the call graph for this function:

6.31.2.70 subroutine MATRIX_VECTOR::VECTOR_DATA_GET_L `(TYPE(VECTOR_TYPE),pointer VECTOR, LOGICAL,dimension(:),pointer DATA,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4384 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_L_TYPE`.

Here is the call graph for this function:

6.31.2.71 subroutine MATRIX_VECTOR::VECTOR_DATA_GET_SP (TYPE(VECTOR_TYPE),pointer VECTOR, REAL(SP),dimension(:),pointer DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a `VECTOR_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4294 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_SP_TYPE`.

Here is the call graph for this function:

6.31.2.72 subroutine MATRIX_VECTOR::VECTOR_DATA_TYPE_SET (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) DATA_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the data type of a vector.

Parameters:

VECTOR A pointer to the vector.
DATA_TYPE The data type to set.

See also:

[MATRIX_VECTOR::DataTypes](#), [MATRIX_VECTOR](#)

ERR The error code

ERROR The error string

Definition at line 4429 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR_DP_TYPE`, `MATRIX_VECTOR_INTG_TYPE`, `MATRIX_VECTOR_L_TYPE`, and `MATRIX_VECTOR_SP_TYPE`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.73 subroutine MATRIX_VECTOR::VECTOR_DESTROY (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Destroys a vector.

Parameters:

VECTOR A pointer to the vector

ERR The error code

ERROR The error string

Definition at line 4475 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `VECTOR_FINALISE()`.

Here is the call graph for this function:

6.31.2.74 subroutine MATRIX_VECTOR::VECTOR_DUPLICATE (TYPE(VECTOR_TYPE),pointer VECTOR, TYPE(VECTOR_TYPE),pointer NEW_VECTOR, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Duplicates a vector structure and returns a pointer to the new vector in `NEW_VECTOR`.

Parameters:

VECTOR A pointer to the vector to duplicate

NEW_VECTOR On return a pointer to the new duplicated vector

ERR The error code

ERROR The error string

Definition at line 4503 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `VECTOR_CREATE_FINISH()`, `VECTOR_CREATE_START()`, `VECTOR_DATA_TYPE_SET()`, `VECTOR_FINALISE()`, and `VECTOR_SIZE_SET()`.

Here is the call graph for this function:

**6.31.2.75 subroutine MATRIX_VECTOR::VECTOR_FINALISE (TYPE(VECTOR_-
TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Finalises a vector and deallocates all memory.

Parameters:

VECTOR A pointer to the vector to finalise

ERR The error code

ERROR The error string

Definition at line 4540 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by VECTOR_CREATE_START(), VECTOR_DESTROY(), and VECTOR_DUPLICATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.76 subroutine MATRIX_VECTOR::VECTOR_INITIALISE (TYPE(VECTOR_-
TYPE),pointer VECTOR, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Initialises a vector.

Parameters:

VECTOR A pointer to the vector

ERR The error code

ERROR The error string

Definition at line 4570 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by VECTOR_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.31.2.77 subroutine MATRIX_VECTOR::VECTOR_SIZE_SET (TYPE(VECTOR_-
TYPE),pointer VECTOR, INTEGER(INTG),intent(in) N, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Sets/changes the size of a vector.

Parameters:

VECTOR A pointer to the vector

N The size of the vector to set

ERR The error code

ERROR The error string

Definition at line 4603 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `VECTOR_DUPLICATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.2.78 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_DP
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDICES, REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the values in a double precision real vector at the indices specified.

Parameters:

VECTOR A pointer to the vector

INDICES `INDICES(i)`. The i'th index to get

VALUES `VALUES(i)`. On return the i'th value to get

ERR The error code

ERROR The error string

Definition at line 4853 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.79 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_DP1
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a double precision vector at the location specified by the index.

Parameters:

VECTOR A pointer to the vector

INDEX The index of the vector to get

VALUE On return the value of the vector at the specified index

ERR The error code

ERROR The error string

Definition at line 4911 of file matrix_vector.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MATRIX_VECTOR_DP_TYPE`.

Here is the call graph for this function:

6.31.2.80 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_INTG (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Gets the values in an integer vector at the indices specified.

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4643 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.81 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Gets a value in an integer vector at the location specified by the index.

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 4701 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.82 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_L (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Gets the values in a logical real vector at the indices specified.

Parameters:

VECTOR A pointer to the vector

INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4958 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.83 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_L1
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets a value in a logical vector at the location specified by the index.

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 5016 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.84 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_SP
`(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Gets the values in a single precision real vector at the indices specified.

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4748 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.85 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET_SP1
 (**TYPE(VECTOR_TYPE)**,pointer **VECTOR**, **INTEGER(INTG)**,intent(in)
INDEX, **REAL(SP)**,intent(out) **VALUE**, **INTEGER(INTG)**,intent(out) **ERR**,
TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Gets a value in a single precision vector at the location specified by the index.

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 4806 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.86 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_DP
 (**TYPE(VECTOR_TYPE)**,pointer **VECTOR**, **INTEGER(INTG)**,dimension(:),intent(in)
INDICES, **REAL(DP)**,dimension(:),intent(in) **VALUES**, **INTEGER(INTG)**,intent(out)
ERR, **TYPE(VARYING_STRING)**,intent(out) **ERROR**, *)

Sets the values in a double precision vector at the specified indices.

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5273 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.87 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_DP1
 (**TYPE(VECTOR_TYPE)**,pointer **VECTOR**, **INTEGER(INTG)**,intent(in)
INDEX, **REAL(DP)**,intent(in) **VALUE**, **INTEGER(INTG)**,intent(out) **ERR**,
TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Sets a value in a double precision vector at the specified index.

Parameters:

VECTOR A pointer to the vector

INDEX The index to set

VALUE The value to set at the specified index

ERR The error code

ERROR The error string

Definition at line 5331 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_DP_TYPE.

Here is the call graph for this function:

6.31.2.88 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_INTG (TYPE(VECTOR_-TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets the values in an integer vector at the specified indices.

Parameters:

VECTOR A pointer to the vector

INDICES INDICES(i). The i'th index to set

VALUES VALUES(i). The i'th value to set

ERR The error code

ERROR The error string

Definition at line 5063 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.89 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets a value in an integer vector at the specified index.

Parameters:

VECTOR A pointer to the vector

INDEX The index to set

VALUE The value to set at the specified index

ERR The error code

ERROR The error string

Definition at line 5121 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_INTG_TYPE.

Here is the call graph for this function:

6.31.2.90 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_L (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets the values in a logical vector at the specified indices.

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5378 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.91 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_L1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets a value in a logical vector at the specified index.

Parameters:

VECTOR A pointer to the vector
INDEX The index to set
VALUE The value to set at the specified index
ERR The error code
ERROR The error string

Definition at line 5436 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MATRIX_VECTOR_L_TYPE.

Here is the call graph for this function:

6.31.2.92 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_SP (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets the values in a single precision vector at the specified indices.

Parameters:

VECTOR A pointer to the vector

INDICES INDICES(i). The i'th index to set

VALUES VALUES(i). The i'th value to set

ERR The error code

ERROR The error string

Definition at line 5168 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.2.93 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET_SP1
(TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in)
INDEX, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets a value in a single precision vector at the specified index.

Parameters:

VECTOR A pointer to the vector

INDEX The index to set

VALUE The value to set at the specified index

ERR The error code

ERROR The error string

Definition at line 5226 of file matrix_vector.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and MATRIX_VECTOR_SP_TYPE.

Here is the call graph for this function:

6.31.3 Variable Documentation

6.31.3.1 INTEGER(INTG),save MATRIX_VECTOR::MATRIX_VECTOR_ID = 1

Definition at line 174 of file matrix_vector.f90.

Referenced by MATRIX_CREATE_FINISH(), and VECTOR_CREATE_FINISH().

6.32 MESH_ROUTINES Namespace Reference

This module handles all mesh (node and element) routines.

Classes

- interface [MESH_NUMBER_OF_COMPONENTS_SET](#)
Sets/changes the number of mesh components for a mesh.
- interface [MESH_NUMBER_OF_ELEMENTS_SET](#)
Sets/changes the number of elements for a mesh.

Functions

- subroutine [DECOMPOSITION_CREATE_FINISH](#) (MESH, DECOMPOSITION, ERR, ERROR,*)
Finishes the creation of a domain decomposition on a given mesh.
- subroutine [DECOMPOSITION_CREATE_START](#) (USER_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,*)
Starts the creation of a domain decomposition for a given mesh.
- subroutine [DECOMPOSITION_DESTROY](#) (USER_NUMBER, MESH, ERR, ERROR,*)
Destroys a domain decomposition identified by a user number and deallocates all memory.
- subroutine [DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE](#) (DECOMPOSITION, ERR, ERROR,*)
Calculates the element domains for a decomposition of a mesh.
- subroutine [DECOMPOSITION_ELEMENT_DOMAIN_SET](#) (DECOMPOSITION, GLOBAL_ELEMENT_NUMBER, DOMAIN_NUMBER, ERR, ERROR,*)
Sets the domain for a given element in a decomposition of a mesh.
- subroutine [DECOMPOSITION_MESH_COMPONENT_NUMBER_SET](#) (DECOMPOSITION, MESH_COMPONENT_NUMBER, ERR, ERROR,*)
Sets/changes the mesh component number which will be used for the decomposition of a mesh.
- subroutine [DECOMPOSITION_NUMBER_OF_DOMAINS_SET](#) (DECOMPOSITION, NUMBER_OF_DOMAINS, ERR, ERROR,*)
Sets/changes the number of domains for a decomposition.
- subroutine [DOMAIN_MAPPINGS_NODES_FINALISE](#) (DOMAIN_MAPPINGS, ERR, ERROR,*)
Finalises the node mapping in the given domain mappings.
- subroutine [DOMAIN_MAPPINGS_NODES_INITIALISE](#) (DOMAIN_MAPPINGS, ERR, ERROR,*)
Initialises the node mapping in the given domain mapping.

- subroutine [DOMAIN_TOPOLOGY_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)
Calculates the domain topology.
- subroutine [DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH](#) (DOMAIN, ERR, ERROR,*)
Initialises the local domain topology from the mesh topology.
- subroutine [DOMAIN_TOPOLOGY_DOFS_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)
Finalises the dofs in the given domain topology.
- subroutine [DOMAIN_TOPOLOGY_DOFS_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)
Initialises the dofs data structures for a domain topology.
- subroutine [DOMAIN_TOPOLOGY_ELEMENT_FINALISE](#) (ELEMENT, ERR, ERROR,*)
Finalises the given domain topology element.
- subroutine [DOMAIN_TOPOLOGY_ELEMENT_INITIALISE](#) (ELEMENT, ERR, ERROR,*)
Initialises the given domain topology element.
- subroutine [DOMAIN_TOPOLOGY_ELEMENTS_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)
Finalises the elements in the given domain topology.
- subroutine [DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)
Initialises the element data structures for a domain topology.
- subroutine [DOMAIN_TOPOLOGY_FINALISE](#) (DOMAIN, ERR, ERROR,*)
Finalises the topology in the given domain.
- subroutine [DOMAIN_TOPOLOGY_INITIALISE](#) (DOMAIN, ERR, ERROR,*)
Initialises the topology for a given domain.
- subroutine [DOMAIN_TOPOLOGY_LINE_FINALISE](#) (LINE, ERR, ERROR,*)
Finalises a line in the given domain topology and deallocates all memory.
- subroutine [DOMAIN_TOPOLOGY_LINE_INITIALISE](#) (LINE, ERR, ERROR,*)
Initialises the line data structure for a domain topology line.
- subroutine [DOMAIN_TOPOLOGY_LINES_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)
Finalises the lines in the given domain topology.
- subroutine [DOMAIN_TOPOLOGY_LINES_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)
Initialises the line data structures for a domain topology.
- subroutine [DOMAIN_TOPOLOGY_NODE_FINALISE](#) (NODE, ERR, ERROR,*)
Finalises the given domain topology node and deallocates all memory.
- subroutine [DOMAIN_TOPOLOGY_NODE_INITIALISE](#) (NODE, ERR, ERROR,*)
Initialises the given domain topology node.
- subroutine [DOMAIN_TOPOLOGY_NODES_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)

Finalises the nodees in the given domain topology.

- subroutine [DOMAIN_TOPOLOGY_NODES_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)

Initialises the nodes data structures for a domain topology.
- subroutine [DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding a node for a domain.
- subroutine [MESH_CREATE_FINISH](#) (REGION, MESH, ERR, ERROR,*)

Finishes the process of creating a mesh on a region.
- subroutine [MESH_CREATE_REGULAR](#) (USER_NUMBER, REGION, ORIGIN, MAXIMUM_EXTENT, NUMBER_ELEMENTS_XI, BASIS, MESH, ERR, ERROR,*)

Creates the regular mesh with the given USER_NUMBER in the specified REGION. The mesh starts at the ORIGIN(:) and has a maximum extent position of MAXIMUM_EXTENT(:) with the NUMBER_OF_ELEMENTS(:) in each direction. Each element is of the specified BASIS type. A pointer to the finished mesh is returned in MESH.
- subroutine [MESH_CREATE_START](#) (USER_NUMBER, REGION, NUMBER_OF_DIMENSIONS, MESH, ERR, ERROR,*)

Starts the process of creating a mesh defined by a user number with the specified NUMBER_OF_DIMENSIONS in the region identified by REGION.
- subroutine [MESH_DESTROY](#) (USER_NUMBER, REGION, ERR, ERROR,*)

Destroys the mesh identified by a user number on the given region and deallocates all memory.
- subroutine [MESH_FINALISE](#) (MESH, ERR, ERROR,*)

Finalises a mesh and deallocates all memory.
- subroutine [MESH_INITIALISE](#) (MESH, ERR, ERROR,*)

Initialises a mesh.
- subroutine [MESH_NUMBER_OF_COMPONENTS_SET_NUMBER](#) (USER_NUMBER, REGION, NUMBER_OF_COMPONENTS, ERR, ERROR,*)

Changes/sets the number of mesh components for a mesh identified by a given user number on a region.
- subroutine [MESH_NUMBER_OF_COMPONENTS_SET_PTR](#) (MESH, NUMBER_OF_COMPONENTS, ERR, ERROR,*)

Changes/sets the number of mesh components for a mesh identified by a pointer.
- subroutine [MESH_NUMBER_OF_ELEMENTS_SET_NUMBER](#) (USER_NUMBER, REGION, NUMBER_OF_ELEMENTS, ERR, ERROR,*)

Changes/sets the number of elements for a mesh identified by a given user number on a region.
- subroutine [MESH_NUMBER_OF_ELEMENTS_SET_PTR](#) (MESH, NUMBER_OF_ELEMENTS, ERR, ERROR,*)

Changes/sets the number of elements for a mesh identified by a pointer.
- subroutine [MESH_TOPOLOGY_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the mesh topology.

- subroutine [MESH_TOPOLOGY_DOFS_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the degrees-of-freedom for a mesh topology.
- subroutine [MESH_TOPOLOGY_DOFS_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)

Finalises the dof data structures for a mesh topology and deallocates any memory.
- subroutine [MESH_TOPOLOGY_DOFS_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)

Initialises the dofs in a given mesh topology.
- subroutine [MESH_TOPOLOGY_ELEMENTS_BASIS_SET](#) (GLOBAL_NUMBER, ELEMENTS, BASIS, ERR, ERROR,*)

Changes/sets the basis for a global element identified by a given global number.
- subroutine [MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH](#) (MESH, MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Finishes the process of creating elements for a specified mesh component in a mesh topology.
- subroutine [MESH_TOPOLOGY_ELEMENTS_CREATE_START](#) (MESH, MESH_COMPONENT_NUMBER, BASIS, ELEMENTS, ERR, ERROR,*)

Starts the process of creating elements in the mesh component identified by MESH and component_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH_ELEMENTS data structure.
- subroutine [MESH_TOPOLOGY_ELEMENTS_DESTROY](#) (TOPOLOGY, ERR, ERROR,*)

Destroys the elements in a mesh topology.
- subroutine [MESH_TOPOLOGY_ELEMENT_FINALISE](#) (ELEMENT, ERR, ERROR,*)

Finalises the given mesh topology element.
- subroutine [MESH_TOPOLOGY_ELEMENT_INITIALISE](#) (ELEMENT, ERR, ERROR,*)

Initialises the given mesh topology element.
- subroutine [MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET](#) (GLOBAL_NUMBER, ELEMENTS, BASIS, ERR, ERROR,*)

Changes/sets the basis for a mesh element identified by a given global number.
- subroutine [MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET](#) (GLOBAL_NUMBER, ELEMENTS, USER_ELEMENT_NODES, ERR, ERROR,*)

Changes/sets the element nodes for a mesh element identified by a given global number.
- subroutine [MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding an element in a mesh topology.
- subroutine [MESH_TOPOLOGY_ELEMENTS_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)

Finalises the elements data structures for a mesh topology and deallocates any memory.
- subroutine [MESH_TOPOLOGY_ELEMENTS_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)

Initialises the elements in a given mesh topology.

- subroutine [MESH_TOPOLOGY_ELEMENTS_NUMBER_SET](#) (GLOBAL_NUMBER, USER_NUMBER, ELEMENTS, ERR, ERROR,*)

Changes/sets the user number for a global element identified by a given global number.
- subroutine [MESH_TOPOLOGY_FINALISE](#) (MESH, ERR, ERROR,*)

Finalises the topology in the given mesh.
- subroutine [MESH_TOPOLOGY_INITIALISE](#) (MESH, ERR, ERROR,*)

Initialises the topology for a given mesh.
- subroutine [MESH_TOPOLOGY_NODE_FINALISE](#) (NODE, ERR, ERROR,*)

Finalises the given mesh topology node.
- subroutine [MESH_TOPOLOGY_NODE_INITIALISE](#) (NODE, ERR, ERROR,*)

Initialises the given mesh topology node.
- subroutine [MESH_TOPOLOGY_NODES_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the nodes used the mesh identified by a given mesh topology.
- subroutine [MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the number of derivatives at each node in a topology.
- subroutine [MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE](#) (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding a node for a mesh.
- subroutine [MESH_TOPOLOGY_NODES_FINALISE](#) (TOPOLOGY, ERR, ERROR,*)

Finalises the nodes data structures for a mesh topology and deallocates any memory.
- subroutine [MESH_TOPOLOGY_NODES_INITIALISE](#) (TOPOLOGY, ERR, ERROR,*)

Initialises the nodes in a given mesh topology.
- subroutine [MESH_USER_NUMBER_FIND](#) (USER_NUMBER, REGION, MESH, ERR, ERROR,*)

Finds and returns in MESH a pointer to the mesh identified by USER_NUMBER in the given REGION. If no mesh with that number exists MESH is left nullified.
- subroutine [MESHES_FINALISE](#) (REGION, ERR, ERROR,*)

Finalises the meshes in the given region.
- subroutine [MESHES_INITIALISE](#) (REGION, ERR, ERROR,*)

Initialises the meshes for the given region.

Variables

- INTEGER(INTG), parameter [DECOMPOSITION_ALL_TYPE](#) = 1

The decomposition contains all elements.

- INTEGER(INTG), parameter **DECOMPOSITION_CALCULATED_TYPE** = 2

The element decomposition is calculated by graph partitioning.

- INTEGER(INTG), parameter **DECOMPOSITION_USER_DEFINED_TYPE** = 3

The user will set the element decomposition.

6.32.1 Detailed Description

This module handles all mesh (node and element) routines.

6.32.2 Function Documentation

6.32.2.1 subroutine MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH (TYPE(MESH_TYPE),pointer *MESH*, TYPE(DECOMPOSITION_TYPE),pointer *DECOMPOSITION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finishes the creation of a domain decomposition on a given mesh.

Parameters:

MESH A pointer to the mesh to decompose.

DECOMPOSITION A pointer to the decomposition to finish creating

ERR The error code

ERROR The error string

Definition at line 115 of file mesh_routines.f90.

References DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE(), BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD_IO_ROUTINES::FIE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.2 subroutine MESH_ROUTINES::DECOMPOSITION_CREATE_START (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(MESH_TYPE),pointer *MESH*, TYPE(DECOMPOSITION_TYPE),pointer *DECOMPOSITION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Starts the creation of a domain decomposition for a given mesh.

Parameters:

USER_NUMBER The user number of the decomposition

MESH A pointer to the mesh to decompose

DECOMPOSITION On return a pointer to the created decomposition. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 185 of file mesh_routines.f90.

References DECOMPOSITION_ALL_TYPE, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by FIELD_IO_ROUTINES::FIE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.3 subroutine MESH_ROUTINES::DECOMPOSITION_DESTROY (INTEGER(INTG),intent(in) **USER_NUMBER**, TYPE(MESH_TYPE),pointer **MESH**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Destroys a domain decomposition identified by a user number and deallocates all memory.

Parameters:

USER_NUMBER The user number of the decomposition to destroy.

MESH A pointer to the mesh containing the decomposition.

ERR The error code

ERROR The error string

Definition at line 276 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.4 subroutine MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_- CALCULATE (TYPE(DECOMPOSITION_TYPE),pointer **DECOMPOSITION**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Calculates the element domains for a decomposition of a mesh.

Parameters:

DECOMPOSITION A pointer to the decomposition to calculate the element domains for.

ERR The error code

ERROR The error string

Definition at line 364 of file mesh_routines.f90.

References KINDS::SP, COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT, COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET(), COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET(), DECOMPOSITION_ALL_TYPE, DECOMPOSITION_CALCULATED_TYPE, DECOMPOSITION_USER_DEFINED_TYPE, BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), KINDS::INTG, CMISS_MPI::MPI_ERROR_CHECK(), CMISS_PARMETIS::PARMETIS_PARTMESHKWAY(), and KINDS::PTR.

Referenced by DECOMPOSITION_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.5 subroutine MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_SET
`(TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,
 INTEGER(INTG),intent(in) GLOBAL_ELEMENT_NUMBER,
 INTEGER(INTG),intent(in) DOMAIN_NUMBER, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Sets the domain for a given element in a decomposition of a mesh.

Parameters:

DECOMPOSITION A pointer to the decomposition to set the element domain for

GLOBAL_ELEMENT_NUMBER The global element number to set the domain for.

DOMAIN_NUMBER The domain of the global element.

ERR The error code

ERROR The error string

Definition at line 552 of file mesh_routines.f90.

References COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.32.2.6 subroutine MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_SET
`(TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,
 INTEGER(INTG),intent(in) MESH_COMPONENT_NUMBER,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
 [private]`

Sets/changes the mesh component number which will be used for the decomposition of a mesh.

Parameters:

DECOMPOSITION A pointer to the decomposition to set the mesh component for

MESH_COMPONENT_NUMBER The mesh component number to set

ERR The error code

ERROR The error string

Definition at line 617 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.32.2.7 subroutine MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET
`(TYPE(DECOMPOSITION_TYPE),pointer DECOMPOSITION,`
`INTEGER(INTG),intent(in) NUMBER_OF_DOMAINS, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Sets/changes the number of domains for a decomposition.

Parameters:

DECOMPOSITION A pointer to the decomposition to set the number of domains for.

NUMBER_OF_DOMAINS The number of domains to set.

ERR The error code

ERROR The error string

Definition at line 662 of file mesh_routines.f90.

References `KINDS::DP`, `COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET()`, `COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET()`, `DECOMPOSITION_ALL_TYPE`, `DECOMPOSITION_CALCULATED_TYPE`, `DECOMPOSITION_DESTROY()`, `DECOMPOSITION_USER_DEFINED_TYPE`, `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::DIAGNOSTICS2`, `DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY`, `DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST`, `DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL`, `DOMAIN_MAPPINGS::DOMAIN_LOCAL_FROM_GLOBAL_CALCULATE()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPING_FINALISE()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_INITIALISE()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_NODES_FINALISE()`, `DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_NODES_INITIALISE()`, `DOMAIN_TOPOLOGY_FINALISE()`, `DOMAIN_TOPOLOGY_INITIALISE()`, `DOMAIN_TOPOLOGY_LINE_INITIALISE()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `KINDS::INTG`, `LISTS::LIST_CREATE_FINISH()`, `LISTS::LIST_CREATE_START()`, `LISTS::LIST_DATA_TYPE_SET()`, `LISTS::LIST_DESTROY()`, `LISTS::LIST_INITIAL_SIZE_SET()`, `LISTS::LIST_INTG_TYPE`, `LISTS::LIST_REMOVE_DUPLICATES()`, and `KINDS::PTR`.

Referenced by `FIELD_IO_ROUTINES::FIE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.8 subroutine MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE
`(TYPE(DOMAIN_MAPPINGS_TYPE),pointer DOMAIN_MAPPINGS,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Finalises the node mapping in the given domain mappings.

Todo

pass in the nodes mapping

Parameters:

DOMAIN_MAPPINGS A pointer to the domain mapping to finalise the nodes for

ERR The error code

ERROR The error string

Definition at line 3011 of file mesh_routines.f90.

References DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.9 subroutine MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE
(TYPE(DOMAIN_MAPPINGS_TYPE),pointer DOMAIN_MAPPINGS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]**

Initialises the node mapping in the given domain mapping.

Todo

finalise on error

Parameters:

DOMAIN_MAPPINGS A pointer to the domain mappings to initialise the nodes for

ERR The error code

ERROR The error string

Definition at line 3043 of file mesh_routines.f90.

References DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_INITIALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.10 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_CALCULATE
(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Calculates the domain topology.

Parameters:

TOPOLOGY A pointer to the domain topology to calculate.

ERR The error code

ERROR The error string

Definition at line 3079 of file mesh_routines.f90.

References DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE(),
BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.11 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE
(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Finalises the dofs in the given domain topology.

Todo

pass in the dofs topolgy

Parameters:

TOPOLOGY A pointer to the domain topology to finalise the dofs for

ERR The error code

ERROR The error string

Definition at line 3298 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.12 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE
(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Initialises the dofs data structures for a domain topology.

Todo

finalise on exit

Parameters:

TOPOLOGY A pointer to the domain topology to initialise the dofs for

ERR The error code

ERROR The error string

Definition at line 3330 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.13 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENT_FINALISE
`(TYPE(DOMAIN_ELEMENT_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`**

Finalises the given domain topology element.

Parameters:

ELEMENT The domain element to finalise

ERR The error code

ERROR The error string

Definition at line 3366 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_ELEMENTS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.14 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENT_INITIALISE
`(TYPE(DOMAIN_ELEMENT_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`**

Initialises the given domain topology element.

Parameters:

ELEMENT The domain element to initialise

ERR The error code

ERROR The error string

Definition at line 3391 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.15 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-
FINALISE (TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Finalises the elements in the given domain topology.

Todo

pass in the domain elements

Parameters:

TOPOLOGY A pointer to the domain topology to finalise the elements for

ERR The error code

ERROR The error string

Definition at line 3417 of file mesh_routines.f90.

References DOMAIN_TOPOLOGY_ELEMENT_FINALISE(), BASE_ROUTINES::ENTERS(),
BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.16 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_-
INITIALISE (TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Initialises the element data structures for a domain topology.

Todo

finalise on error

Parameters:

TOPOLOGY A pointer to the domain topology to initialise the elements for

ERR The error code

ERROR The error string

Definition at line 3453 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.17 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE
(TYPE(DOMAIN_TYPE),pointer DOMAIN, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalises the topology in the given domain.

Todo

pass in domain topology

Parameters:

DOMAIN A pointer to the domain to finalise the topology for
ERR The error code
ERROR The error string

Definition at line 3491 of file mesh_routines.f90.

References DOMAIN_TOPOLOGY_DOFS_FINALISE(), DOMAIN_TOPOLOGY_ELEMENTS_FINALISE(), DOMAIN_TOPOLOGY_LINES_FINALISE(), DOMAIN_TOPOLOGY_NODES_FINALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.18 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE
(TYPE(DOMAIN_TYPE),pointer DOMAIN, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises the topology for a given domain.

Todo

finalise on error

Parameters:

ERR The error code
ERROR The error string

Definition at line 3524 of file mesh_routines.f90.

References DOMAIN_TOPOLOGY_CALCULATE(), DOMAIN_TOPOLOGY_DOFS_INITIALISE(), DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE(), DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH(), DOMAIN_TOPOLOGY_LINES_INITIALISE(), DOMAIN_TOPOLOGY_NODES_INITIALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DECOMPOSITION_NUMBER_OF_DOMAINS_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.19 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH
(TYPE(DOMAIN_TYPE),pointer DOMAIN, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises the local domain topology from the mesh topology.

Parameters:

DOMAIN A pointer to the domain to initialise the domain topology from the mesh topology for
ERR The error code
ERROR The error string

Definition at line 3125 of file mesh_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-
ROUTINES::DIAGNOSTICS1, DOMAIN_TOPOLOGY_ELEMENT_INITIALISE(), DOMAIN_-
TOPOLOGY_NODE_INITIALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), and KINDS::PTR.

Referenced by DOMAIN_TOPOLOGY_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.20 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE
(TYPE(DOMAIN_LINE_TYPE) LINE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalises a line in the given domain topology and deallocates all memory.

Parameters:

LINE The domain line to finalise
ERR The error code
ERROR The error string

Definition at line 3572 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-
ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_LINES_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.21 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE
(TYPE(DOMAIN_LINE_TYPE) LINE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises the line data structure for a domain topology line.

Parameters:

LINE The domain line to initialise

ERR The error code

ERROR The error string

Definition at line 3600 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DECOMPOSITION_NUMBER_OF_DOMAINS_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.22 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_FINALISE
`(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Finalises the lines in the given domain topology.

Todo

pass in domain lines

Parameters:

TOPOLOGY A pointer to the domain topology to finalise the lines for

ERR The error code

ERROR The error string

Definition at line 3625 of file mesh_routines.f90.

References `DOMAIN_TOPOLOGY_LINE_FINALISE()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.23 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_INITIALISE
`(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Initialises the line data structures for a domain topology.

Todo

finalise on error

Parameters:

TOPOLOGY A pointer to the domain topology to initialise the lines for

ERR The error code

ERROR The error string

Definition at line 3661 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.24 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE
(TYPE(DOMAIN_NODE_TYPE) NODE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalises the given domain topology node and deallocates all memory.

Parameters:

NODE The domain node to finalise

ERR The error code

ERROR The error string

Definition at line 3696 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_NODES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.25 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE
(TYPE(DOMAIN_NODE_TYPE) NODE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises the given domain topology node.

Parameters:

NODE The domain node to initialise

ERR The error code

ERROR The error string

Definition at line 3723 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.26 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_FINALISE
(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 \ast) [private]

Finalises the nodees in the given domain topology.

Todo

pass in domain nodes

Parameters:

TOPOLOGY A pointer to the domain topology to initialise the nodes for

ERR The error code

ERROR The error string

Definition at line 3752 of file mesh_routines.f90.

References DOMAIN_TOPOLOGY_NODE_FINALISE(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.27 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_INITIALISE
(TYPE(DOMAIN_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 \ast) [private]

Initialises the nodes data structures for a domain topology.

Todo

finalise on error

Parameters:

TOPOLOGY A pointer to the domain topology to initialise the nodes for

ERR The error code

ERROR The error string

Definition at line 3788 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by DOMAIN_TOPOLOGY_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.28 subroutine MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE (TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TOPOLOGY, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Calculates the element numbers surrounding a node for a domain.

Parameters:

TOPOLOGY A pointer to the domain topology to calculate the elements surrounding the nodes for

ERR The error code

ERROR The error string

Definition at line 3826 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Referenced by **DOMAIN_TOPOLOGY_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.29 subroutine MESH_ROUTINES::MESH_CREATE_FINISH (TYPE(REGION_TYPE),pointer REGION, TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Finishes the process of creating a mesh on a region.

Todo

remove region

Parameters:

REGION A pointer to the region containing the mesh

MESH A pointer to the mesh to finish creating

ERR The error code

ERROR The error string

Definition at line 3915 of file mesh_routines.f90.

References **BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE**, **BASE_ROUTINES::DIAGNOSTICS1**, **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **MESH_TOPOLOGY_CALCULATE()**, and **KINDS::PTR**.

Referenced by **FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()**, and **MESH_CREATE_REGULAR()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.30 subroutine MESH_ROUTINES::MESH_CREATE_REGULAR
(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, REAL(DP),dimension(:),intent(in) ORIGIN, REAL(DP),dimension(:),intent(in) MAXIMUM_EXTENT, INTEGER(INTG),dimension(:),intent(in) NUMBER_ELEMENTS_XI, TYPE(BASIS_TYPE),pointer BASIS, TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Creates the regular mesh with the given USER_NUMBER in the specified REGION. The mesh starts at the ORIGIN(:) and has a maximum extent position of MAXIMUM_EXTENT(:) with the NUMBER_OF_ELEMENTS(:) in each direction. Each element is of the specified BASIS type. A pointer to the finished mesh is returned in MESH.

Todo

move to generated mesh routines

Parameters:

USER_NUMBER The user number of the mesh to create

REGION A pointer to the region containing the mesh

ORIGIN ORIGIN(i). ORIGIN(i) contains the i'th coordinate in the region of the origin of the regular mesh.

MAXIMUM_EXTENT MAXIMUM_EXTENT(i). MAXIMUM_EXTENT(i) contains the i'th extent (or size) of the regular mesh.

NUMBER_ELEMENTS_XI NUMBER_ELEMENTS_XI(i). NUMBER_ELEMENTS_XI(i) contains the number of elements in the i'th direction

BASIS A pointer to the basis to use for each element in the regular mesh

MESH On return, a pointer to the generated mesh

ERR The error code

ERROR The error string

Definition at line 4008 of file mesh_routines.f90.

References KINDS::_DP, COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE, BASE_ROUTINES::ENTER(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MESH_CREATE_FINISH(), MESH_CREATE_START(), MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH(), MESH_TOPOLOGY_ELEMENTS_CREATE_START(), MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET(), NODE_ROUTINES::NODE_INITIAL_POSITION_SET(), NODE_ROUTINES::NODES_CREATE_FINISH(), and NODE_ROUTINES::NODES_CREATE_START().

Here is the call graph for this function:

6.32.2.31 subroutine MESH_ROUTINES::MESH_CREATE_START
(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(in) NUMBER_OF_DIMENSIONS, TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Starts the process of creating a mesh defined by a user number with the specified NUMBER_OF_DIMENSIONS in the region identified by REGION.

Parameters:

USER_NUMBER The user number of the mesh to create
REGION A pointer to the region to create the mesh on
NUMBER_OF_DIMENSIONS The number of dimensions in the mesh.
MESH On exit, a pointer to the created mesh. Must not be associated on entry.
ERR The error code
ERROR The error string

Definition at line 4186 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_INITIALISE()`, `MESH_TOPOLOGY_INITIALISE()`, `MESH_USER_NUMBER_FIND()`, and `KINDS::PTR`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`, and `MESH_CREATE_REGULAR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.32 subroutine MESH_ROUTINES::MESH_DESTROY (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Destroys the mesh identified by a user number on the given region and deallocates all memory.

Parameters:

USER_NUMBER The user number of the mesh to destroy
REGION A pointer to the region containing the mesh
ERR The error code
ERROR The error string

Definition at line 4282 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_FINALISE()`, and `KINDS::PTR`.

Referenced by `MESSES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.33 subroutine MESH_ROUTINES::MESH_FINALISE (TYPE(MESH_TYPE),pointer *MESH*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises a mesh and deallocates all memory.

Parameters:

MESH A pointer to the mesh to finalise

ERR The error code

ERROR The error string

Definition at line 4366 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `MESH_TOPOLOGY_FINALISE()`.

Referenced by `MESH_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.34 subroutine `MESH_ROUTINES::MESH_INITIALISE` (`TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Initialises a mesh.

Parameters:

MESH A pointer to the mesh to initialise

ERR The error code

ERROR The error string

Definition at line 4395 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.35 subroutine `MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_NUMBER` (`INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Changes/sets the number of mesh components for a mesh identified by a given user number on a region.

Parameters:

USER_NUMBER The user number of the mesh to set the number of components for

REGION A pointer to the region containing the mesh

NUMBER_OF_COMPONENTS The number of components to set for the mesh

ERR The error code

ERROR The error string

Definition at line 4438 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_NUMBER_OF_COMPONENTS_SET_PTR()`, and `MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

```
6.32.2.36 subroutine MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR
  (TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(in)
   NUMBER_OF_COMPONENTS, INTEGER(INTG),intent(out) ERR,
   TYPE(VARYING_STRING),intent(out) ERROR, */ [private]
```

Changes/sets the number of mesh components for a mesh identified by a pointer.

Parameters:

MESH A pointer to the mesh to set the number of components for

NUMBER_OF_COMPONENTS The number of components to set.

ERR The error code

ERROR The error string

Definition at line 4476 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_ELEMENTS_INITIALISE()`, `MESH_TOPOLOGY_NODES_INITIALISE()`, and `KINDS::PTR`.

Referenced by `MESH_NUMBER_OF_COMPONENTS_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.32.2.37 subroutine MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER
  (INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer
   REGION, INTEGER(INTG),intent(in) NUMBER_OF_ELEMENTS,
   INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
   */ [private]
```

Changes/sets the number of elements for a mesh identified by a given user number on a region.

Parameters:

USER_NUMBER The user number of the mesh to set the number of elements for

REGION A pointer to the region containing the mesh

NUMBER_OF_ELEMENTS The number of elements to set

ERR The error code

ERROR The error string

Definition at line 4546 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_NUMBER_OF_ELEMENTS_SET_PTR()`, and `MESH_USER_NUMBER_FIND()`.

Here is the call graph for this function:

**6.32.2.38 subroutine MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_PTR
(TYPE(MESH_TYPE),pointer *MESH*, INTEGER(INTG),intent(in)
NUMBER_OF_ELEMENTS, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Changes/sets the number of elements for a mesh identified by a pointer.

Parameters:

MESH A pointer to the mesh to set the number of elements for

NUMBER_OF_ELEMENTS The number of elements to set

ERR The error code

ERROR The error string

Definition at line 4584 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **KINDS::PTR**.

Referenced by **MESH_NUMBER_OF_ELEMENTS_SET_NUMBER()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.39 subroutine MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Calculates the mesh topology.

Parameters:

TOPOLOGY A pointer to the mesh topology to calculate

ERR The error code

ERROR The error string

Definition at line 4644 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **MESH_TOPOLOGY_DOFS_CALCULATE()**, **MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE()**, **MESH_TOPOLOGY_NODES_CALCULATE()**, **MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE()**, and **MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE()**.

Referenced by **MESH_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.40 subroutine MESH_ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Calculates the degrees-of-freedom for a mesh topology.

Parameters:

TOPOLOGY A pointer to the mesh topology to calculate the dofs for

ERR The error code

ERROR The error string

Definition at line 4681 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_CALCULATE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.41 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_DOFS_FINALISE`(*TYPE(MESH_TOPOLOGY_TYPE)*,pointer *TOPOLOGY***,
INTEGER(INTG,intent(out) ERR, ***TYPE(VARYING_STRING),intent(out) ERROR***,
*) [private]**

Finalises the dof data structures for a mesh topology and deallocates any memory.

Todo

pass in dofs

Parameters:

TOPOLOGY A pointer to the mesh topology to finalise the dofs for

ERR The error code

ERROR The error string

Definition at line 4727 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.42 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE`(*TYPE(MESH_TOPOLOGY_TYPE)*,pointer *TOPOLOGY***,
INTEGER(INTG,intent(out) ERR, ***TYPE(VARYING_STRING),intent(out) ERROR***,
*) [private]**

Initialises the dofs in a given mesh topology.

Todo

finalise on error

Parameters:

TOPOLOGY A pointer to the mesh topology to initialise the dofs for
ERR The error code
ERROR The error string

Definition at line 4757 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.43 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE`
`(TYPE(MESH_ELEMENT_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises the given mesh topology element.

Parameters:

ELEMENT The mesh element to finalise
ERR The error code
ERROR The error string

Definition at line 5048 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_ELEMENTS_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.44 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_INITIALISE`
`(TYPE(MESH_ELEMENT_TYPE) ELEMENT, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the given mesh topology element.

Parameters:

ELEMENT The mesh element to finalise
ERR The error code
ERROR The error string

Definition at line 5075 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by MESH_TOPOLOGY_ELEMENTS_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.32.2.45 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_-
ELEMENTS_CALCULATE (TYPE(MESH_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast)$  [private]
```

Calculates the element numbers surrounding an element in a mesh topology.

Parameters:

TOPOLOGY A pointer to the mesh topology to calculate the elements adjacent to elements for

ERR The error code

ERROR The error string

Definition at line 5274 of file mesh_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INTG_TYPE, LISTS::LIST_REMOVE_DUPLICATES(), and KINDS::PTR.

Referenced by MESH_TOPOLOGY_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.32.2.46 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET
(INTEGER(INTG),intent(in) GLOBAL_NUMBER, TYPE(MESH_-
ELEMENTS_TYPE),pointer ELEMENTS, TYPE(BASIS_TYPE),pointer BASIS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast)$  [private]
```

Changes/sets the basis for a global element identified by a given global number.

Todo

use user number.

Parameters:

GLOBAL_NUMBER The global number of the element to set the basis for

ELEMENTS A pointer to the elements to set the basis for before element number?

BASIS A pointer to the basis to set. allow number version as well.

ERR The error code

ERROR The error string

Definition at line 4792 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.32.2.47 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH` (TYPE(MESH_TYPE),pointer *MESH*, INTEGER(INTG),intent(in) *MESH_COMPONENT_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finishes the process of creating elements for a specified mesh component in a mesh topology.

Parameters:

MESH A pointer to the mesh to finish creating the elements for

MESH_COMPONENT_NUMBER The mesh component number to finish creating the elements for

ERR The error code

ERROR The error string

Definition at line 4847 of file `mesh_routines.f90`.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`, and `MESH_CREATE_REGULAR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.48 subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START` (TYPE(MESH_TYPE),pointer *MESH*, INTEGER(INTG),intent(in) *MESH_COMPONENT_NUMBER*, TYPE(BASIS_TYPE),pointer *BASIS*, TYPE(MESH_ELEMENTS_TYPE),pointer *ELEMENTS*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Starts the process of creating elements in the mesh component identified by *MESH* and *component_idx*. The elements will be created with a default basis of *BASIS*. *ELEMENTS* is the returned pointer to the *MESH_ELEMENTS* data structure.

Parameters:

MESH A pointer to the mesh to start creating the elements on

MESH_COMPONENT_NUMBER The mesh component number

BASIS A pointer to the default basis to use

ELEMENTS On return, a pointer to the created mesh elements

ERR The error code

ERROR The error string

Definition at line 4926 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_ELEMENT_INITIALISE()`, `MESH_TOPOLOGY_ELEMENTS_FINALISE()`, and `KINDS::PTR`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`, and `MESH_CREATE_REGULAR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.49 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) *ERROR*,
**) [private]*

Destroys the elements in a mesh topology.

Todo

as this is a user routine it should take a mesh pointer like create start and finish? Split this into destroy and finalise?

Parameters:

TOPOLOGY A pointer to the mesh topology to destroy the elements for

ERR The error code

ERROR The error string

Definition at line 5004 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.32.2.50 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET (*INTEGER(INTG),intent(in) GLOBAL_NUMBER*,
TYPE(MESH_ELEMENTS_TYPE),pointer ELEMENTS,
TYPE(BASIS_TYPE),pointer BASIS, *INTEGER(INTG),intent(out) ERR*,
TYPE(VARYING_STRING),intent(out) ERROR, **)*

Changes/sets the basis for a mesh element identified by a given global number.

Todo

should take user number

Parameters:

GLOBAL_NUMBER The global number of the element to set the basis for

ELEMENTS A pointer to the elements to set the basis for before number?

BASIS A pointer to the basis to set

ERR The error code

ERROR The error string

Definition at line 5101 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.51 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS - ELEMENT_NODES_SET (INTEGER(INTG),intent(in) *GLOBAL_NUMBER*, TYPE(MESH_ELEMENTS_TYPE),pointer *ELEMENTS*, INTEGER(INTG),dimension(:),intent(in) *USER_ELEMENT_NODES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Changes/sets the element nodes for a mesh element identified by a given global number.

Todo

specify by user number not global number

Parameters:

GLOBAL_NUMBER The global number of the element to set the nodes for

ELEMENTS A pointer to the elements to set before number?

USER_ELEMENT_NODES *USER_ELEMENT_NODES*(i). *USER_ELEMENT_NODES*(i) is the i'th user node number for the element

ERR The error code

ERROR The error string

Definition at line 5177 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `NODE_ROUTINES::NODE_CHECK_EXISTS()`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`, and `MESH_CREATE_REGULAR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.52 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS - FINALISE (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises the elements data structures for a mesh topology and deallocates any memory.

Todo

pass in elements

Parameters:

TOPOLOGY A pointer to the mesh topology to finalise the elements for
ERR The error code
ERROR The error string

Definition at line 5486 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and MESH_TOPOLOGY_ELEMENT_FINALISE().

Referenced by MESH_TOPOLOGY_ELEMENTS_CREATE_START(), and MESH_TOPOLOGY_-FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.53 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_-INITIALISE (TYPE(MESH_TOPOLOGY_TYPE),pointer TOPOLOGY, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Initialises the elements in a given mesh topology.

Todo

finalise on error

Parameters:

TOPOLOGY A pointer to the mesh topology to initialise the elements for
ERR The error code
ERROR The error string

Definition at line 5520 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by MESH_NUMBER_OF_COMPONENTS_SET_PTR(), and MESH_TOPOLOGY_-INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.54 subroutine MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET (INTEGER(INTG),intent(in) GLOBAL_NUMBER, INTEGER(INTG),intent(in) USER_NUMBER, TYPE(MESH_ELEMENTS_TYPE),pointer ELEMENTS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Changes/sets the user number for a global element identified by a given global number.

Todo

Check that the user number doesn't already exist.

Parameters:

GLOBAL_NUMBER The global number of the elements to set.

USER_NUMBER The user number of the element to set

ELEMENTS A pointer to the elements to set the user number for This should be the first parameter.

ERR The error code

ERROR The error string

Definition at line 5556 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.55 subroutine MESH_ROUTINES::MESH_TOPOLOGY_FINALISE
`(TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalises the topology in the given mesh.

Todo

pass in the mesh topology

Parameters:

MESH A pointer to the mesh to finalise the topology for

ERR The error code

ERROR The error string

Definition at line 5598 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_DOFS_FINALISE()`, `MESH_TOPOLOGY_ELEMENTS_FINALISE()`, `MESH_TOPOLOGY_NODES_FINALISE()`, and `KINDS::PTR`.

Referenced by `MESH_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.56 subroutine MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE
`(TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises the topology for a given mesh.

Todo

finalise on error

Parameters:

MESH A pointer to the mesh to initialise the mesh topology for

ERR The error code

ERROR The error string

Definition at line 5634 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MESH_TOPOLOGY_DOFS_INITIALISE()`, `MESH_TOPOLOGY_ELEMENTS_INITIALISE()`, `MESH_TOPOLOGY_NODES_INITIALISE()`, and `KINDS::PTR`.

Referenced by `MESH_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.57 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODE_FINALISE
(TYPE(MESH_NODE_TYPE) NODE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalises the given mesh topology node.

Parameters:

NODE The mesh node to finalise

ERR The error code

ERROR The error string

Definition at line 5681 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `MESH_TOPOLOGY_NODES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.58 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE
(TYPE(MESH_NODE_TYPE) NODE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises the given mesh topology node.

Parameters:

NODE The mesh node to initialise

ERR The error code

ERROR The error string

Definition at line 5707 of file mesh_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by MESH_TOPOLOGY_NODES_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.59 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODES_-
CALCULATE (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Calculates the nodes used the mesh identified by a given mesh topology.

Parameters:

TOPOLOGY A pointer to the mesh topology

ERR The error code

ERROR The error string

Definition at line 5735 of file mesh_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), MESH_TOPOLOGY_NODE_INITIALISE(), TREES::TREE_CREATE_FINISH(), TREES::TREE_CREATE_START(), TREES::TREE_DESTROY(), TREES::TREE_DETACH_AND_DESTROY(), TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_INSERT(), and TREES::TREE_NO_DUPLICATES_ALLOWED.

Referenced by MESH_TOPOLOGY_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.60 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_-
CALCULATE (TYPE(MESH_TOPOLOGY_TYPE),pointer *TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Calculates the number of derivatives at each node in a topology.

Parameters:

TOPOLOGY A pointer to the mesh topology to calculate the derivates at each node for

ERR The error code

ERROR The error string

Definition at line 5839 of file mesh_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), LISTS::LIST_CREATE_FINISH(), LISTS::LIST_CREATE_START(), LISTS::LIST_DATA_TYPE_SET(), LISTS::LIST_DESTROY(), LISTS::LIST_INITIAL_SIZE_SET(), LISTS::LIST_INTG_TYPE, and LISTS::LIST_REMOVE_DUPLICATES().

Referenced by MESH_TOPOLOGY_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.32.2.61 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODES_-
FINALISE (TYPE(MESH_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast)$  [private]
```

Finalises the nodes data structures for a mesh topology and deallocates any memory.

Todo

pass in nodes

Parameters:

TOPOLOGY A pointer to the mesh topology to finalise the nodes for
ERR The error code
ERROR The error string

Definition at line 6029 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_-ROUTINES::EXITS()**, and **MESH_TOPOLOGY_NODE_FINALISE()**.

Referenced by **MESH_TOPOLOGY_FINALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

```
6.32.2.62 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODES_-
INITIALISE (TYPE(MESH_TOPOLOGY_TYPE),pointer TOPOLOGY,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 $\ast)$  [private]
```

Initialises the nodes in a given mesh topology.

Todo

finalise on errors

Parameters:

TOPOLOGY A pointer to the mesh topology to initialise the nodes for
ERR The error code
ERROR The error string

Definition at line 6064 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_-ROUTINES::EXITS()**.

Referenced by **MESH_NUMBER_OF_COMPONENTS_SET_PTR()**, and **MESH_TOPOLOGY_-INITIALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.63 subroutine MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE (*TYPE(MESH_TOPOLOGY_TYPE)*,*pointer TOPOLOGY*,
INTEGER(INTG),intent(out) *ERR*, *TYPE(VARYING_STRING),intent(out)* *ERROR*,
**) [private]*

Calculates the element numbers surrounding a node for a mesh.

Parameters:

TOPOLOGY A pointer to the mesh topology to calculate the elements surrounding each node for

ERR The error code

ERROR The error string

Definition at line 5945 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Referenced by **MESH_TOPOLOGY_CALCULATE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.32.2.64 subroutine MESH_ROUTINES::MESH_USER_NUMBER_FIND
(INTEGER(INTG),intent(in) ***USER_NUMBER***, *TYPE(REGION_TYPE)*,*pointer REGION*, *TYPE(MESH_TYPE)*,*pointer MESH*, *INTEGER(INTG),intent(out)* *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, **) [private]*

Finds and returns in **MESH** a pointer to the mesh identified by **USER_NUMBER** in the given **REGION**. If no mesh with that number exists **MESH** is left nullified.

Parameters:

USER_NUMBER The user number of the mesh to find

REGION The region containing the mesh

MESH On return, a pointer to the mesh of the specified user number. In no mesh with the specified user number exists the pointer is returned NULL.

ERR The error code

ERROR The error string

Definition at line 6100 of file mesh_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **KINDS::PTR**.

Referenced by **MESH_CREATE_START()**, **MESH_NUMBER_OF_COMPONENTS_SET_NUMBER()**, and **MESH_NUMBER_OF_ELEMENTS_SET_NUMBER()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.65 subroutine MESH_ROUTINES::MESSES_FINALISE (TYPE(REGION_-
TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Finalises the meshes in the given region.

Todo

pass in meshes

Parameters:

REGION A pointer to the region to finalise the meshes for.

ERR The error code

ERROR The error string

Definition at line 6146 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), MESH_DESTROY(), and KINDS::PTR.

Referenced by REGION_ROUTINES::REGION_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.32.2.66 subroutine MESH_ROUTINES::MESSES_INITIALISE (TYPE(REGION_-
TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Initialises the meshes for the given region.

Todo

finalise on error

Parameters:

REGION A pointer to the region to initialise the meshes for.

ERR The error code

ERROR The error string

Definition at line 6182 of file mesh_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by REGION_ROUTINES::REGION_CREATE_START(), and REGION_-ROUTINES::REGION_SUB_REGION_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33 NODE_ROUTINES Namespace Reference

This module handles all node routines.

Functions

- subroutine `NODE_CHECK_EXISTS` (`USER_NUMBER`, `REGION`, `NODE_EXISTS`, `GLOBAL_NUMBER`, `ERR`, `ERROR,*`)
- subroutine `NODE_DESTROY` (`NODE`, `ERR`, `ERROR,*`)
- subroutine `NODE_INITIAL_POSITION_SET` (`GLOBAL_NUMBER`, `INITIAL_POSITION`, `NODES`, `ERR`, `ERROR,*`)
- subroutine `NODE_NUMBER_SET` (`GLOBAL_NUMBER`, `USER_NUMBER`, `NODES`, `ERR`, `ERROR,*`)
- subroutine `NODES_CREATE_FINISH` (`REGION`, `ERR`, `ERROR,*`)
- subroutine `NODES_CREATE_START` (`NUMBER_OF_NODES`, `REGION`, `NODES`, `ERR`, `ERROR,*`)
- subroutine `NODES_FINALISE` (`REGION`, `ERR`, `ERROR,*`)
- subroutine `NODES_INITIALISE` (`REGION`, `ERR`, `ERROR,*`)

6.33.1 Detailed Description

This module handles all node routines.

6.33.2 Function Documentation

6.33.2.1 subroutine NODE_ROUTINES::NODE_CHECK_EXISTS (INTEGER(INTG) *USER_NUMBER*, TYPE(REGION_TYPE),pointer *REGION*, LOGICAL,intent(out) *NODE_EXISTS*, INTEGER(INTG),intent(out) *GLOBAL_NUMBER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 75 of file node_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `TREES::TREE_NODE_VALUE_GET()`, and `TREES::TREE_SEARCH()`.

Referenced by `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.2 subroutine NODE_ROUTINES::NODE_DESTROY (TYPE(NODE_TYPE) *NODE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 126 of file node_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `NODES_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.3 subroutine NODE_ROUTINES::NODE_INITIAL_POSITION_SET
`(INTEGER(INTG),intent(in) GLOBAL_NUMBER, REAL(DP),dimension(:),intent(in)`
`INITIAL_POSITION, TYPE(NODES_TYPE),pointer NODES,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Definition at line 153 of file node_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by MESH_ROUTINES::MESH_CREATE_REGULAR().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.4 subroutine NODE_ROUTINES::NODE_NUMBER_SET `(INTEGER(INTG),intent(in)`
`GLOBAL_NUMBER, INTEGER(INTG),intent(in) USER_NUMBER,`
`TYPE(NODES_TYPE),pointer NODES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 216 of file node_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), TREES::TREE_ITEM_DELETE(), TREES::TREE_ITEM_INSERT(), and TREES::TREE_NODE_INSERT_SUCESSFUL.

Referenced by FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.5 subroutine NODE_ROUTINES::NODES_CREATE_FINISH
`(TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 267 of file node_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and TREES::TREE_OUTPUT().

Referenced by FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(), and MESH_ROUTINES::MESH_CREATE_REGULAR().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.6 subroutine NODE_ROUTINES::NODES_CREATE_START
`(INTEGER(INTG),intent(in) NUMBER_OF_NODES, TYPE(REGION_TYPE),pointer`
`REGION, TYPE(NODES_TYPE),pointer NODES, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 322 of file node_routines.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), TREES::TREE_CREATE_FINISH(), TREES::TREE_CREATE_START(), TREES::TREE_INSERT_TYPE_SET(), TREES::TREE_ITEM_INSERT(), and TREES::TREE_NO_DUPLICATES_ALLOWED.

Referenced by FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH(), and MESH_ROUTINES::MESH_CREATE_REGULAR().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.7 subroutine NODE_ROUTINES::NODES_FINALISE (TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 404 of file node_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), NODE_DESTROY(), and TREES::TREE_DESTROY().

Referenced by REGION_ROUTINES::REGION_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

6.33.2.8 subroutine NODE_ROUTINES::NODES_INITIALISE (TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 443 of file node_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by REGION_ROUTINES::REGION_CREATE_START(), and REGION_ROUTINES::REGION_SUB_REGION_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.34 PROBLEM_CONSTANTS Namespace Reference

This module handles all problem wide constants.

Variables

- INTEGER(INTG), parameter `PROBLEM_NO_CLASS` = 0
- INTEGER(INTG), parameter `PROBLEM_ELASTICITY_CLASS` = 1
- INTEGER(INTG), parameter `PROBLEM_FLUID_MECHANICS_CLASS` = 2
- INTEGER(INTG), parameter `PROBLEM_ELECTROMAGNETICS_CLASS` = 3
- INTEGER(INTG), parameter `PROBLEM_CLASSICAL_FIELD_CLASS` = 4
- INTEGER(INTG), parameter `PROBLEM_MODAL_CLASS` = 5
- INTEGER(INTG), parameter `PROBLEM_FITTING_CLASS` = 6
- INTEGER(INTG), parameter `PROBLEM_OPTIMISATION_CLASS` = 7
- INTEGER(INTG), parameter `PROBLEM_NO_TYPE` = 0
- INTEGER(INTG), parameter `PROBLEM_LINEAR_ELASTICITY_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEMFINITE_ELASTICITY_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_STOKES_FLUID_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_NAVIER_STOKES_FLUID_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_ELECTROSTATIC_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_MAGNETOSTATIC_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_MAXWELLS_EQUATIONS_TYPE` = 3
- INTEGER(INTG), parameter `PROBLEM_LAPLACE_EQUATION_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_POISSON_EQUATION_TYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_HELMHOLTZ_EQUATION_TYPE` = 3
- INTEGER(INTG), parameter `PROBLEM_WAVE_EQUATION_TYPE` = 4
- INTEGER(INTG), parameter `PROBLEM_DIFFUSION_EQUATION_TYPE` = 5
- INTEGER(INTG), parameter `PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE` = 6
- INTEGER(INTG), parameter `PROBLEMREACTION_DIFFUSION_EQUATION_TYPE` = 7
- INTEGER(INTG), parameter `PROBLEM_BIHAMONIC_EQUATION_TYPE` = 8
- INTEGER(INTG), parameter `PROBLEM_LINEAR_ELASTIC_MODAL_TYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_NO_SUBTYPE` = 0
- INTEGER(INTG), parameter `PROBLEM_STANDARD_LAPLACE_SUBTYPE` = 1
- INTEGER(INTG), parameter `PROBLEM_GENERALISED_LAPLACE_SUBTYPE` = 2
- INTEGER(INTG), parameter `PROBLEM_SETUP_INITIAL_TYPE` = 1

Initial setup for a problem.

- INTEGER(INTG), parameter `PROBLEM_SETUP_CONTROL_TYPE` = 2

Solver setup for a problem.

- INTEGER(INTG), parameter `PROBLEM_SETUP SOLUTION_TYPE` = 3

Solution parameters setup for a problem.

- INTEGER(INTG), parameter `PROBLEM_SETUP_SOLVER_TYPE` = 4

Solver setup for a problem.

- INTEGER(INTG), parameter `PROBLEM_SETUP_START_ACTION` = 1

Start setup action.

- INTEGER(INTG), parameter **PROBLEM_SETUP_FINISH_ACTION** = 2
Finish setup action.
- INTEGER(INTG), parameter **PROBLEM_SETUP_DO_ACTION** = 3
Do setup action.
- INTEGER(INTG), parameter **PROBLEM_SOLUTION_NO_OUTPUT** = 0
No output.
- INTEGER(INTG), parameter **PROBLEM_SOLUTION_TIMING_OUTPUT** = 1
Timing information output.
- INTEGER(INTG), parameter **PROBLEM_SOLUTION_MATRIX_OUTPUT** = 2
All below and solution matrices output.
- INTEGER(INTG), parameter **PROBLEM_SOLVER_NO_OUTPUT** = 0
No output.
- INTEGER(INTG), parameter **PROBLEM_SOLVER_TIMING_OUTPUT** = 1
Timing information output.
- INTEGER(INTG), parameter **PROBLEM_SOLVER_SOLVER_OUTPUT** = 2
All below and solver output.
- INTEGER(INTG), parameter **PROBLEM_SOLVER_SPARSE_MATRICES** = 1
Use sparse solver matrices.
- INTEGER(INTG), parameter **PROBLEM_SOLVER_FULL_MATRICES** = 2
Use fully populated solver matrices.

6.34.1 Detailed Description

This module handles all problem wide constants.

6.34.2 Variable Documentation

6.34.2.1 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_ADVECTION_DIFFUSION_EQUATION_TYPE = 6

Definition at line 80 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

6.34.2.2 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_BIHAMONIC_EQUATION_TYPE = 8

Definition at line 82 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

6.34.2.3 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS = 4

Definition at line 57 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_CREATE_START(), PROBLEM_ROUTINES::PROBLEM_SETUP(), and PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

6.34.2.4 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_EQUATION_TYPE = 5

Definition at line 79 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

6.34.2.5 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_ELASTICITY_CLASS = 1

Definition at line 54 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SETUP(), and PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

6.34.2.6 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_ELECTROMAGNETICS_CLASS = 3

Definition at line 56 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SETUP(), and PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

6.34.2.7 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_ELECTROSTATIC_TYPE = 1

Definition at line 71 of file problem_constants.f90.

6.34.2.8 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEMFINITE_ELASTICITY_TYPE = 2

Definition at line 66 of file problem_constants.f90.

6.34.2.9 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_FITTING_CLASS = 6

Definition at line 59 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

**6.34.2.10 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_FLUID_-
MECHANICS_CLASS = 2**

Definition at line 55 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SETUP(), and PROBLEM_-ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

**6.34.2.11 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_-
GENERALISED_LAPLACE_SUBTYPE = 2**

Definition at line 94 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP(), and LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET().

**6.34.2.12 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_-
HELMHOLTZ_EQUATION_TYPE = 3**

Definition at line 77 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

**6.34.2.13 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_LAPLACE_-
EQUATION_TYPE = 1**

Definition at line 75 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP(), and PROBLEM_ROUTINES::PROBLEM_CREATE_START().

**6.34.2.14 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_LINEAR_-
ELASTIC_MODAL_TYPE = 1**

Definition at line 84 of file problem_constants.f90.

**6.34.2.15 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_LINEAR_-
ELASTICITY_TYPE = 1**

Definition at line 65 of file problem_constants.f90.

**6.34.2.16 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_-
MAGNETOSTATIC_TYPE = 2**

Definition at line 72 of file problem_constants.f90.

6.34.2.17 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_MAXWELLS_EQUATIONS_TYPE = 3

Definition at line 73 of file problem_constants.f90.

6.34.2.18 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_MODAL_CLASS = 5

Definition at line 58 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SETUP(), and PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

6.34.2.19 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_NAVIER_STOKES_FLUID_TYPE = 2

Definition at line 69 of file problem_constants.f90.

6.34.2.20 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_NO_CLASS = 0

Definition at line 53 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_INITIALISE().

6.34.2.21 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_NO_SUBTYPE = 0

Definition at line 87 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_INITIALISE().

6.34.2.22 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_NO_TYPE = 0

Definition at line 63 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_INITIALISE().

6.34.2.23 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_OPTIMISATION_CLASS = 7

Definition at line 60 of file problem_constants.f90.

Referenced by PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR().

6.34.2.24 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE = 2

Definition at line 76 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

**6.34.2.25 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEMREACTION_-
DIFFUSION_EQUATION_TYPE = 7**

Definition at line 81 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_-
SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

**6.34.2.26 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_STANDARD_-
LAPLACE_SUBTYPE = 1**

Definition at line 93 of file problem_constants.f90.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_STANDARD_SETUP(),
LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET(), and
PROBLEM_ROUTINES::PROBLEM_CREATE_START().

**6.34.2.27 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_STOKES_-
FLUID_TYPE = 1**

Definition at line 68 of file problem_constants.f90.

**6.34.2.28 INTEGER(INTG),parameter PROBLEM_CONSTANTS::PROBLEM_WAVE_-
EQUATION_TYPE = 4**

Definition at line 78 of file problem_constants.f90.

Referenced by CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_-
SET(), and CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP().

6.35 PROBLEM_ROUTINES Namespace Reference

This module handles all problem routines.

Classes

- interface [PROBLEM_DESTROY](#)
- interface [PROBLEM_SPECIFICATION_SET](#)

Functions

- subroutine [PROBLEM_CREATE_FINISH](#) (PROBLEM, ERR, ERROR,*)
Finishes the process of creating a problem.
- subroutine [PROBLEM_CREATE_START](#) (USER_NUMBER, PROBLEM, ERR, ERROR,*)
Starts the process of creating a problem defined by USER_NUMBER.
- subroutine [PROBLEM_DESTROY_NUMBER](#) (USER_NUMBER, ERR, ERROR,*)
Destroys a problem identified by a user number.
- subroutine [PROBLEM_DESTROY_PTR](#) (PROBLEM, ERR, ERROR,*)
Destroys a problem identified by a pointer.
- subroutine [PROBLEM_FINALISE](#) (PROBLEM, ERR, ERROR,*)
Finalise the problem and deallocate all memory.
- subroutine [PROBLEM_INITIALISE](#) (PROBLEM, ERR, ERROR,*)
Initialises a problem.
- subroutine [PROBLEM_CONTROL_CREATE_FINISH](#) (PROBLEM, ERR, ERROR,*)
Finish the creation of the control for the problem.
- subroutine [PROBLEM_CONTROL_CREATE_START](#) (PROBLEM, ERR, ERROR,*)
Start the creation of a problem control for a problem.
- subroutine [PROBLEM_CONTROL_DESTROY](#) (PROBLEM, ERR, ERROR,*)
Destroy the control for a problem.
- subroutine [PROBLEM_CONTROL_FINALISE](#) (CONTROL, ERR, ERROR,*)
Finalise the control for a problem and deallocate all memory.
- subroutine [PROBLEM_CONTROL_INITIALISE](#) (PROBLEM, ERR, ERROR,*)
Initialise the control for a problem.
- subroutine [PROBLEM_SETUP](#) (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)
Sets up the specifics for a problem.
- subroutine [PROBLEM_SOLVE](#) (PROBLEM, ERR, ERROR,*)
Solves a problem.

- subroutine [PROBLEM_SOLUTION_solve](#) (SOLUTION, ERR, ERROR,*)

Solves a solution.
- subroutine [PROBLEM_SOLUTIONS_createFinish](#) (PROBLEM, ERR, ERROR,*)

Finish the creation of solutions for a problem.
- subroutine [PROBLEM_SOLUTIONS_createStart](#) (PROBLEM, ERR, ERROR,*)

Start the creation of a solution for the problem.
- subroutine [PROBLEM_SOLUTION_EQUATIONS_SET_ADD](#) (PROBLEM, SOLUTION_INDEX, EQUATIONS_SET, EQUATIONS_SET_INDEX, ERR, ERROR,*)

Adds an equations set to a problem solution.
- subroutine [PROBLEM_SOLUTION_finalise](#) (SOLUTION, ERR, ERROR,*)

Finalises a solution and deallocates all memory.
- subroutine [PROBLEM_SOLUTION_initialise](#) (SOLUTION, ERR, ERROR,*)

Initialises the solution for a problem.
- subroutine [PROBLEM_SOLUTIONS_finalise](#) (PROBLEM, ERR, ERROR,*)

Finalises the solutions for a problem and deallocates all memory.
- subroutine [PROBLEM_SOLUTIONS_initialise](#) (PROBLEM, ERR, ERROR,*)

Initialises the solutions for a problem.
- subroutine [PROBLEM_SOLVER_CREATE_FINISH](#) (PROBLEM, ERR, ERROR,*)

Finish the creation of the solver for the problem solutions.
- subroutine [PROBLEM_SOLVER_CREATE_START](#) (PROBLEM, ERR, ERROR,*)

Start the creation of a problem solver for a problem.
- subroutine [PROBLEM_SOLVER_DESTROY](#) (PROBLEM, ERR, ERROR,*)

Destroy the solvers for a problem.
- subroutine [PROBLEM_SOLVER_GET](#) (PROBLEM, SOLUTION_INDEX, SOLVER, ERR, ERROR,*)

Returns a pointer to the solver for a solution on a problem.
- subroutine [PROBLEM_SPECIFICATION_SET_NUMBER](#) (USER_NUMBER, PROBLEM_CLASS, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.
- subroutine [PROBLEM_SPECIFICATION_SET_PTR](#) (PROBLEM, PROBLEM_CLASS, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.
- subroutine [PROBLEM_USER_NUMBER_FIND](#) (USER_NUMBER, PROBLEM, ERR, ERROR,*)

Finds and returns in PROBLEM a pointer to the problem identified by USER_NUMBER. If no problem with that USER_NUMBER exists PROBLEM is left nullified.

- subroutine **PROBLEMS_FINALISE** (ERR, ERROR,*)
Finalises all problems and deallocates all memory.
- subroutine **PROBLEMS_INITIALISE** (ERR, ERROR,*)
Initialises all problems.

Variables

- INTEGER(INTG), parameter **NUMBER_OF_PROBLEM_LINEARITIES** = 3
The number of problem linearity types defined.
- INTEGER(INTG), parameter **PROBLEM_LINEAR** = 1
The problem is linear.
- INTEGER(INTG), parameter **PROBLEM_NONLINEAR** = 2
The problem is non-linear.
- INTEGER(INTG), parameter **PROBLEM_NONLINEAR_BCS** = 3
The problem has non-linear boundary conditions.
- INTEGER(INTG), parameter **NUMBER_OF_PROBLEM_TIME_TYPES** = 3
The number of problem time dependence types defined.
- INTEGER(INTG), parameter **PROBLEM_STATIC** = 1
The problem is static and has no time dependence.
- INTEGER(INTG), parameter **PROBLEM_DYNAMIC** = 2
The problem is dynamic.
- INTEGER(INTG), parameter **PROBLEM_QUASISTATIC** = 3
The problem is quasi-static.
- TYPE(**PROBLEMS_TYPE**), target **PROBLEMS**

6.35.1 Detailed Description

This module handles all problem routines.

6.35.2 Function Documentation

6.35.2.1 subroutine **PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH** (**TYPE(PROBLEM_TYPE)**,pointer **PROBLEM**, **INTEGER(INTG)**,**intent(out)** **ERR**, **TYPE(VARYING_STRING)**,**intent(out)** **ERROR**, *)

Finish the creation of the control for the problem.

Parameters:

PROBLEM A pointer to the problem to finish the control for

ERR The error code

ERROR The error string

Definition at line 429 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`.

Here is the call graph for this function:

6.35.2.2 subroutine `PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START` `(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, *)`

Start the creation of a problem control for a problem.

Parameters:

PROBLEM A pointer to the problem to start the creation of a control for.

ERR The error code

ERROR The error string

Definition at line 468 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_CONTROL_FINALISE()`, `PROBLEM_CONTROL_INITIALISE()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`.

Here is the call graph for this function:

6.35.2.3 subroutine `PROBLEM_ROUTINES::PROBLEM_CONTROL_DESTROY` `(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, *)`

Destroy the control for a problem.

Parameters:

PROBLEM A pointer to the problem to destroy the control for.

ERR The error code

ERROR The error string

Definition at line 506 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `PROBLEM_CONTROL_FINALISE()`.

Here is the call graph for this function:

6.35.2.4 subroutine PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE
 (**TYPE(PROBLEM_CONTROL_TYPE),pointer** *CONTROL*,
INTEGER(INTG),intent(out) *ERR*, **TYPE(VARYING_STRING),intent(out)** *ERROR*, *)
 [private]

Finalise the control for a problem and deallocate all memory.

Parameters:

CONTROL A pointer to the problem control to finalise.

ERR The error code

ERROR The error string

Definition at line 538 of file problem_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Referenced by **PROBLEM_CONTROL_CREATE_START()**, **PROBLEM_CONTROL_DESTROY()**, **PROBLEM_CONTROL_INITIALISE()**, and **PROBLEM_FINALISE()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.5 subroutine PROBLEM_ROUTINES::PROBLEM_CONTROL_INITIALISE
 (**TYPE(PROBLEM_TYPE),pointer** *PROBLEM*, **INTEGER(INTG),intent(out)** *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialise the control for a problem.

Parameters:

PROBLEM A pointer to the problem to initialise the control for.

ERR The error code

ERROR The error string

Definition at line 563 of file problem_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **PROBLEM_CONTROL_FINALISE()**.

Referenced by **PROBLEM_CONTROL_CREATE_START()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.6 subroutine PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH
 (**TYPE(PROBLEM_TYPE),pointer** *PROBLEM*, **INTEGER(INTG),intent(out)** *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finishes the process of creating a problem.

Parameters:

PROBLEM A pointer to the problem to finish creating.

ERR The error code

ERROR The error string

Definition at line 132 of file problem_routines.f90.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-
ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
BASE_ROUTINES::EXITS(), PROBLEM_SETUP(), PROBLEM_CONSTANTS::PROBLEM_SETUP_-
FINISH_ACTION, PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE, PROBLEM_-
SOLUTIONS_INITIALISE(), PROBLEMS, and KINDS::PTR.`

Here is the call graph for this function:

**6.35.2.7 subroutine PROBLEM_ROUTINES::PROBLEM_CREATE_START
(INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(PROBLEM_TYPE),pointer
PROBLEM, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Starts the process of creating a problem defined by *USER_NUMBER*.

Parameters:

USER_NUMBER The user number of the problem to create

PROBLEM On return, a pointer to the created problem. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 184 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
ROUTINES::EXITS(), PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS,
PROBLEM_INITIALISE(), PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE,
PROBLEM_SETUP(), PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE, PROBLEM_-
CONSTANTS::PROBLEM_SETUP_START_ACTION, PROBLEM_CONSTANTS::PROBLEM_-
STANDARD_LAPLACE_SUBTYPE, PROBLEM_USER_NUMBER_FIND(), PROBLEMS, and
KINDS::PTR.`

Here is the call graph for this function:

**6.35.2.8 subroutine PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER
(INTEGER(INTG),intent(in) *USER_NUMBER*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Destroys a problem identified by a user number.

Parameters:

USER_NUMBER The user number of the problem to destroy

ERR The error code

ERROR The error string

Definition at line 253 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEMS`, and `KINDS::PTR`.

Here is the call graph for this function:

6.35.2.9 subroutine PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Destroys a problem identified by a pointer.

Parameters:

PROBLEM A pointer to the problem to destroy

ERR The error code

ERROR The error string

Definition at line 304 of file `problem_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_FINALISE()`, `PROBLEMS`, and `KINDS::PTR`.

Here is the call graph for this function:

6.35.2.10 subroutine PROBLEM_ROUTINES::PROBLEM_FINALISE
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Finalise the problem and deallocate all memory.

Parameters:

PROBLEM A pointer to the problem to finalise.

ERR The error code

ERROR The error string

Definition at line 366 of file `problem_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_CONTROL_FINALISE()`, and `PROBLEM_SOLUTIONS_FINALISE()`.

Referenced by `PROBLEM_DESTROY_PTR()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.11 subroutine PROBLEM_ROUTINES::PROBLEM_INITIALISE
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialises a problem.

Parameters:

PROBLEM The pointer to the problem

ERR The error code !<The error code

ERROR The error string !<The error string

Definition at line 394 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_CONSTANTS::PROBLEM_NO_CLASS`, `PROBLEM_CONSTANTS::PROBLEM_NO_SUBTYPE`, `PROBLEM_CONSTANTS::PROBLEM_NO_TYPE`, and `PROBLEMS`.

Referenced by `PROBLEM_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.12 subroutine PROBLEM_ROUTINES::PROBLEM_SETUP (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(in) *SETUP_TYPE*, INTEGER(INTG),intent(in) *ACTION_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets up the specifics for a problem.

Parameters:

PROBLEM A pointer to the problem to setup

SETUP_TYPE The problem setup type

See also:

[PROBLEM_CONSTANTS::SetupTypes](#),[PROBLEM_CONSTANTS](#)

ACTION_TYPE The problem setup action type

See also:

[PROBLEM_CONSTANTS::SetupActionTypes](#),[CONSTANTS_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 601 of file problem_routines.f90.

References `CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP()`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS`, `PROBLEM_CONSTANTS::PROBLEM_ELASTICITY_CLASS`, `PROBLEM_CONSTANTS::PROBLEM_ELECTROMAGNETICS_CLASS`, `PROBLEM_CONSTANTS::PROBLEM_FLUID_MECHANICS_CLASS`, and `PROBLEM_CONSTANTS::PROBLEM_MODAL_CLASS`.

Referenced by `PROBLEM_CONTROL_CREATE_FINISH()`, `PROBLEM_CONTROL_CREATE_START()`, `PROBLEM_CREATE_FINISH()`, `PROBLEM_CREATE_START()`, `PROBLEM_SOLUTION_EQUIATIONS_SET_ADD()`, `PROBLEM_SOLUTIONS_CREATE_FINISH()`, `PROBLEM_SOLUTIONS_CREATE_START()`, `PROBLEM_SOLVER_CREATE_FINISH()`, and `PROBLEM_SOLVER_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.13 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTION_EQUATIONS_SET_-
ADD** (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(in)
SOLUTION_INDEX, TYPE(EQUATIONS_SET_TYPE),pointer
EQUATIONS_SET, INTEGER(INTG),intent(out) *EQUATIONS_SET_INDEX*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Adds an equations set to a problem solution.

Parameters:

PROBLEM A pointer to the problem to add the equations set to a solution
SOLUTION_INDEX The solution index in the problem to add the equations set to
EQUATIONS_SET A pointer to the equations set to add
EQUATIONS_SET_INDEX On return, the index of the equations set that has been added.
ERR The error code
ERROR The error string

Definition at line 833 of file problem_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), PROBLEM_SETUP(), PROBLEM_CONSTANTS::PROBLEM_SETUP_-DO_ACTION, PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_TYPE, and KINDS::PTR.

Here is the call graph for this function:

6.35.2.14 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTION_FINALISE
 (TYPE(SOLUTION_TYPE),pointer *SOLUTION*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises a solution and deallocates all memory.

Parameters:

SOLUTION A pointer to the solution to finalise
ERR The error code
ERROR The error string

Definition at line 898 of file problem_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and SOLVER_ROUTINES::SOLVER_DESTROY().

Referenced by PROBLEM_SOLUTIONS_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.15 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTION_INITIALISE
 (TYPE(SOLUTION_TYPE),pointer *SOLUTION*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialises the solution for a problem.

Parameters:

SOLUTION A pointer to the solution to initialise

ERR The error code

ERROR The error string

Definition at line 926 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `PROBLEM_SOLUTIONS_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.16 subroutine PROBLEM_ROUTINES::PROBLEM SOLUTION SOLVE
 (TYPE(SOLUTION_TYPE),pointer **SOLUTION**, INTEGER(INTG),intent(out) **ERR**,
 TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]**

Solves a solution.

Parameters:

SOLUTION A pointer to the solution to solve

ERR The error code

ERROR The error string

Definition at line 695 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `KINDS::PTR`, `SOLVER_ROUTINES::SOLVER_SOLVE()`, and `SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE()`.

Referenced by `PROBLEM_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.17 subroutine PROBLEM_ROUTINES::PROBLEM SOLUTIONS CREATE FINISH
 (TYPE(PROBLEM_TYPE),pointer **PROBLEM**, INTEGER(INTG),intent(out) **ERR**,
 TYPE(VARYING_STRING),intent(out) **ERROR**, *)**

Finish the creation of solutions for a problem.

Parameters:

PROBLEM A pointer to the problem

ERR The error code

ERROR The error string

Definition at line 756 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_TYPE`, and `KINDS::PTR`.

Here is the call graph for this function:

6.35.2.18 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_CREATE_START (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Start the creation of a solution for the problem.

Todo

Should this return a pointer to the problem solution???

Parameters:

PROBLEM A pointer to the problem to create a solution for

ERR The error code !<The error code

ERROR The error string !<The error string

Definition at line 800 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_TYPE`, and `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`.

Here is the call graph for this function:

6.35.2.19 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_FINALISE (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises the solutions for a problem and deallocates all memory.

Parameters:

PROBLEM A pointer to the problem to finalise the solutions for

ERR The error code

ERROR The error string

Definition at line 960 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SOLUTION_FINALISE()`, and `KINDS::PTR`.

Referenced by `PROBLEM_FINALISE()`, and `PROBLEM_SOLUTIONS_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.20 subroutine PROBLEM_ROUTINES::PROBLEM_SOLUTIONS_INITIALISE
 (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Initialises the solutions for a problem.

Parameters:

PROBLEM A pointer to the problem to initialise the solutions for

ERR The error code

ERROR The error string

Definition at line 995 of file problem_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_-ROUTINES::EXITS()**, **PROBLEM_SOLUTION_INITIALISE()**, **PROBLEM_SOLUTIONS_FINALISE()**, and **KINDS::PTR**.

Referenced by **PROBLEM_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.35.2.21 subroutine PROBLEM_ROUTINES::PROBLEM_SOLVE (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Solves a problem.

Parameters:

PROBLEM A pointer to the problem to solve.

ERR The error code

ERROR The error string

Definition at line 646 of file problem_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_-ROUTINES::EXITS()**, **PROBLEM_SOLUTION_SOLVE()**, and **KINDS::PTR**.

Here is the call graph for this function:

**6.35.2.22 subroutine PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH
 (TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Finish the creation of the solver for the problem solutions.

Parameters:

PROBLEM A pointer to the problem to finish the solvers for

ERR The error code

ERROR The error string

Definition at line 1046 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION`, and `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`.

Here is the call graph for this function:

**6.35.2.23 subroutine PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`**

Start the creation of a problem solver for a problem.

Parameters:

PROBLEM A pointer to the problem to start the creation of a solution for.

ERR The error code

ERROR The error string

Definition at line 1075 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEM_SETUP()`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE`, `PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION`, and `KINDS::PTR`.

Here is the call graph for this function:

**6.35.2.24 subroutine PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)`**

Destroy the solvers for a problem.

Parameters:

PROBLEM A pointer to the problem to destroy the solver for.

ERR The error code

ERROR The error string

Definition at line 1119 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `KINDS::PTR`, and `SOLVER_ROUTINES::SOLVER_DESTROY()`.

Here is the call graph for this function:

**6.35.2.25 subroutine PROBLEM_ROUTINES::PROBLEM_SOLVER_GET
`(TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(in)
 SOLUTION_INDEX, TYPE(SOLVER_TYPE),pointer SOLVER,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`**

Returns a pointer to the solver for a solution on a problem.

Parameters:

PROBLEM A pointer to the problem to get the solver for.

SOLUTION_INDEX The solution index to get the solver for.

SOLVER On return, a pointer to the solver. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 1160 of file problem_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

6.35.2.26 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER
 (INTEGER(INTG),intent(in) **USER_NUMBER**, INTEGER(INTG),intent(in)
PROBLEM_CLASS, INTEGER(INTG),intent(in) **PROBLEM_EQUATION_TYPE**,
 INTEGER(INTG),intent(in) **PROBLEM_SUBTYPE**, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.

Parameters:

USER_NUMBER The user number of the problem to set the specification for.

PROBLEM_CLASS The problem class to set.

PROBLEM_EQUATION_TYPE The problem equation to set.

PROBLEM_SUBTYPE The problem subtype.

ERR The error code

ERROR The error string

Definition at line 1208 of file problem_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and PROBLEM_USER_NUMBER_FIND().

Here is the call graph for this function:

6.35.2.27 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR
 (TYPE(PROBLEM_TYPE),pointer **PROBLEM**, INTEGER(INTG),intent(in)
PROBLEM_CLASS, INTEGER(INTG),intent(in) **PROBLEM_EQUATION_TYPE**,
 INTEGER(INTG),intent(in) **PROBLEM_SUBTYPE**, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.

Parameters:

PROBLEM A pointer to the problem to set the specification for.

PROBLEM_CLASS The problem class to set.

PROBLEM_EQUATION_TYPE The problem equation type to set.

PROBLEM_SUBTYPE The problem subtype to set.

ERR The error code

ERROR The error string

Definition at line 1237 of file problem_routines.f90.

References CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS, PROBLEM_CONSTANTS::PROBLEM_EQUIVALENCE_CLASS, PROBLEM_CONSTANTS::PROBLEM_ELASTICITY_CLASS, PROBLEM_CONSTANTS::PROBLEM_ELECTROMAGNETICS_CLASS, PROBLEM_CONSTANTS::PROBLEM_FITTING_CLASS, PROBLEM_CONSTANTS::PROBLEM_FLUID_MECHANICS_CLASS, PROBLEM_CONSTANTS::PROBLEM_MODAL_CLASS, and PROBLEM_CONSTANTS::PROBLEM_OPTIMISATION_CLASS.

Here is the call graph for this function:

6.35.2.28 subroutine PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND
(INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(PROBLEM_TYPE),pointer *PROBLEM*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finds and returns in *PROBLEM* a pointer to the problem identified by *USER_NUMBER*. If no problem with that *USER_NUMBER* exists *PROBLEM* is left nullified.

Parameters:

USER_NUMBER The user number to find.

PROBLEM On return a pointer to the problem with the given user number. If no problem with that user number exists then the pointer is returned as NULL. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 1291 of file problem_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), PROBLEMS, and KINDS::PTR.

Referenced by PROBLEM_CREATE_START(), and PROBLEM_SPECIFICATION_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.29 subroutine PROBLEM_ROUTINES::PROBLEMS_FINALISE
(INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finalises all problems and deallocates all memory.

Parameters:

ERR The error code

ERROR The error string

Definition at line 1328 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `PROBLEMS`, and `KINDS::PTR`.

Referenced by `CMISS::CMISS_FINALISE()`, and `CMISS::CMISS_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.35.2.30 subroutine `PROBLEM_ROUTINES::PROBLEMS_INITIALISE` (`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`, *)

Initialises all problems.

Parameters:

ERR The error code

ERROR The error string

Definition at line 1355 of file problem_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `PROBLEMS`.

Here is the call graph for this function:

6.35.3 Variable Documentation

6.35.3.1 `TYPE(PROBLEMS_TYPE),target PROBLEM_ROUTINES::PROBLEMS`

Definition at line 93 of file problem_routines.f90.

Referenced by `PROBLEM_CREATE_FINISH()`, `PROBLEM_CREATE_START()`, `PROBLEM_DESTROY_NUMBER()`, `PROBLEM_DESTROY_PTR()`, `PROBLEM_INITIALISE()`, `PROBLEM_USER_NUMBER_FIND()`, `PROBLEMS_FINALISE()`, and `PROBLEMS_INITIALISE()`.

6.36 REGION_ROUTINES Namespace Reference

This module contains all region routines.

Classes

- interface [REGION_COORDINATE_SYSTEM_SET](#)
- interface [REGION_LABEL_SET](#)

Functions

- `TYPE(COORDINATE_SYSTEM_TYPE) REGION_COORDINATE_SYSTEM_GET (REGION, ERR, ERROR)`
- subroutine `REGION_COORDINATE_SYSTEM_SET_NUMBER (USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)`
- subroutine `REGION_COORDINATE_SYSTEM_SET_PTR (REGION, COORDINATE_SYSTEM, ERR, ERROR,*)`
- subroutine `REGION_CREATE_FINISH (REGION, ERR, ERROR,*)`
- subroutine `REGION_CREATE_START (USER_NUMBER, REGION, ERR, ERROR,*)`
- subroutine `REGION_DESTROY (USER_NUMBER, ERR, ERROR,*)`
- `TYPE(VARYING_STRING) REGION_LABEL_GET (REGION, ERR, ERROR)`
- subroutine `REGION_LABEL_SET_NUMBER (USER_NUMBER, LABEL, ERR, ERROR,*)`
- subroutine `REGION_LABEL_SET_PTR (REGION, LABEL, ERR, ERROR,*)`
- subroutine `REGION_SUB_REGION_CREATE_START (USER_NUMBER, PARENT_REGION, SUB_REGION, ERR, ERROR,*)`
- subroutine `REGION_SUB_REGION_CREATE_FINISH (REGION, ERR, ERROR,*)`
- subroutine `REGION_USER_NUMBER_FIND (USER_NUMBER, REGION, ERR, ERROR,*)`
- subroutine `REGION_USER_NUMBER_FIND_PTR (USER_NUMBER, REGION, START_REGION, ERR, ERROR,*)`
- subroutine `REGIONS_INITIALISE (ERR, ERROR,*)`
- subroutine `REGIONS_FINALISE (ERR, ERROR,*)`

Variables

- `TYPE(REGION_TYPE), target GLOBAL_REGION`

6.36.1 Detailed Description

This module contains all region routines.

6.36.2 Function Documentation

6.36.2.1 `TYPE(COORDINATE_SYSTEM_TYPE) REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET (TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)`

Definition at line 92 of file region_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.36.2.2 subroutine REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_NUMBER
`(INTEGER(INTG),intent(in) USER_NUMBER,`
`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Definition at line 135 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `REGION_COORDINATE_SYSTEM_SET_PTR()`, and `REGION_USER_NUMBER_FIND()`.

Here is the call graph for this function:

6.36.2.3 subroutine REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR
`(TYPE(REGION_TYPE),pointer REGION, TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 165 of file `region_routines.f90`.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `REGION_COORDINATE_SYSTEM_SET_NUMBER()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.2.4 subroutine REGION_ROUTINES::REGION_CREATE_FINISH
`(TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 205 of file `region_routines.f90`.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE`, `BASE_ROUTINES::DIAGNOSTICS1`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `GLOBAL_REGION`, and `KINDS::PTR`.

Here is the call graph for this function:

6.36.2.5 subroutine REGION_ROUTINES::REGION_CREATE_START
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 249 of file `region_routines.f90`.

References `COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND()`, `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE()`,

BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELDS_INITIALISE(), GLOBAL_REGION, MESH_ROUTINES::MESHS_INITIALISE(), NODE_ROUTINES::NODES_INITIALISE(), KINDS::PTR, and REGION_USER_NUMBER_FIND().

Here is the call graph for this function:

6.36.2.6 subroutine REGION_ROUTINES::REGION_DESTROY (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 338 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_ROUTINES::FIELDS_FINALISE(), MESH_ROUTINES::MESHS_FINALISE(), NODE_ROUTINES::NODES_FINALISE(), KINDS::PTR, and REGION_USER_NUMBER_FIND().

Referenced by REGIONS_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.2.7 TYPE(VARYING_STRING) REGION_ROUTINES::REGION_LABEL_GET (TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)

Definition at line 418 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.36.2.8 subroutine REGION_ROUTINES::REGION_LABEL_SET_NUMBER (INTEGER(INTG),intent(in) USER_NUMBER, CHARACTER(LEN=*),intent(in) LABEL, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 462 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), REGION_LABEL_SET_PTR(), and REGION_USER_NUMBER_FIND().

Here is the call graph for this function:

6.36.2.9 subroutine REGION_ROUTINES::REGION_LABEL_SET_PTR (TYPE(REGION_TYPE),pointer REGION, CHARACTER(LEN=*),intent(in) LABEL, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 492 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by REGION_LABEL_SET_NUMBER().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.36.2.10 subroutine REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH
 (TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Definition at line 622 of file region_routines.f90.

References BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE, BASE_-
 ROUTINES::DIAGNOSTICS1, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(),
 BASE_ROUTINES::EXITS(), and KINDS::PTR.

Here is the call graph for this function:

**6.36.2.11 subroutine REGION_ROUTINES::REGION_SUB_REGION_CREATE_START
 (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer
PARENT_REGION, TYPE(REGION_TYPE),pointer *SUB_REGION*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)**

Definition at line 528 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), EQUATIONS_SET_ROUTINES::EQUATIONS_-
 SETS_INITIALISE(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), FIELD_-
 ROUTINES::FIELDS_INITIALISE(), MESH_ROUTINES::MESHS_INITIALISE(), NODE_-
 ROUTINES::NODES_INITIALISE(), KINDS::PTR, and REGION_USER_NUMBER_FIND().

Here is the call graph for this function:

**6.36.2.12 subroutine REGION_ROUTINES::REGION_USER_NUMBER_FIND
 (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer
REGION, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Definition at line 667 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-
 ROUTINES::EXITS(), GLOBAL_REGION, KINDS::PTR, and REGION_USER_NUMBER_FIND_-
 PTR().

Referenced by REGION_COORDINATE_SYSTEM_SET_NUMBER(), REGION_CREATE_START(),
 REGION_DESTROY(), REGION_LABEL_SET_NUMBER(), and REGION_SUB_REGION_-
 CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.2.13 subroutine REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR
 (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer
REGION, TYPE(REGION_TYPE),pointer *START_REGION*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Definition at line 706 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and KINDS::PTR.

Referenced by REGION_USER_NUMBER_FIND().

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.2.14 subroutine REGION_ROUTINES::REGIONS_FINALISE
 (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Definition at line 790 of file region_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), GLOBAL_REGION, KINDS::PTR, and REGION_DESTROY().

Referenced by CMISS::CMISS_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.2.15 subroutine REGION_ROUTINES::REGIONS_INITIALISE
 (INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *)

Definition at line 750 of file region_routines.f90.

References COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), and GLOBAL_REGION.

Referenced by CMISS::CMISS_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.36.3 Variable Documentation

6.36.3.1 TYPE(REGION_TYPE),target REGION_ROUTINES::GLOBAL_REGION

Definition at line 67 of file region_routines.f90.

Referenced by REGION_CREATE_FINISH(), REGION_CREATE_START(), REGION_USER_NUMBER_FIND(), REGIONS_FINALISE(), and REGIONS_INITIALISE().

6.37 SOLVER_MATRICES_ROUTINES Namespace Reference

This module handles all solver matrix and rhs routines.

Functions

- subroutine [SOLVER_MATRICES_CREATE_FINISH](#) (SOLVER_MATRICES, ERR, ERROR,*)
Finishes the process of creating the solver matrices.
- subroutine [SOLVER_MATRICES_CREATE_START](#) (SOLVER, SOLVER_MATRICES, ERR, ERROR,*)
Starts the process of creating the solver matrices.
- subroutine [SOLVER_MATRICES_DESTROY](#) (SOLVER_MATRICES, ERR, ERROR,*)
Destroys the solver matrices.
- subroutine [SOLVER_MATRICES_FINALISE](#) (SOLVER_MATRICES, ERR, ERROR,*)
Finalises the solver matrices and deallocates all memory.
- subroutine [SOLVER_MATRICES_INITIALISE](#) (SOLVER, ERR, ERROR,*)
Initialises the solver matrices for a solver.
- subroutine [SOLVER_MATRICES_LIBRARY_TYPE_SET](#) (SOLVER_MATRICES, LIBRARY_TYPE, ERR, ERROR,*)
Sets the library type for the solver matrices (and vectors).
- subroutine [SOLVER_MATRICES_OUTPUT](#) (ID, SOLVER_MATRICES, ERR, ERROR,*)
Outputs the solver matrices.
- subroutine [SOLVER_MATRICES_STORAGE_TYPE_SET](#) (SOLVER_MATRICES, STORAGE_TYPE, ERR, ERROR,*)
Sets the storage type (sparsity) of the solver matrices.
- subroutine [SOLVER_MATRIX_STRUCTURE_CALCULATE](#) (SOLVER_MATRIX, NUMBER_OF_NON_ZEROS, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)
Calculates the structure (sparsity) of the solver matrix from the solution mapping.
- subroutine [SOLVER_MATRIX_FINALISE](#) (SOLVER_MATRIX, ERR, ERROR,*)
Finalises a solver matrix and deallocates all memory.
- subroutine [SOLVER_MATRIX_INITIALISE](#) (SOLVER_MATRICES, MATRIX_NUMBER, ERR, ERROR,*)
Initialises a solver matrix.

6.37.1 Detailed Description

This module handles all solver matrix and rhs routines.

6.37.2 Function Documentation

6.37.2.1 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH (TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Finishes the process of creating the solver matrices.

Parameters:

SOLVER_MATRICES A pointer to the solver matrices

ERR The error code

ERROR The error string

Definition at line 76 of file solver_matrices_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE**, **MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE**, **KINDS::PTR**, **SOLVER_MATRICES_FINALISE()**, and **SOLVER_MATRIX_STRUCTURE_CALCULATE()**.

Referenced by **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()**, and **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.2 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START (TYPE(SOLVER_TYPE),pointer SOLVER, TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Starts the process of creating the solver matrices.

Parameters:

SOLVER A pointer to the solver to create the solver matrices for

SOLVER_MATRICES A pointer to the solver matrices

ERR The error code

ERROR The error string

Definition at line 179 of file solver_matrices_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **SOLVER_MATRICES_FINALISE()**, and **SOLVER_MATRICES_INITIALISE()**.

Referenced by **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()**, and **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.3 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY
 (TYPE(SOLVER_MATRICES_TYPE),pointer *SOLVER_MATRICES*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Destroy the solver matrices.

Parameters:

SOLVER_MATRICES A pointer the solver matrices to destroy
ERR The error code
ERROR The error string

Definition at line 222 of file solver_matrices_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and SOLVER_MATRICES_FINALISE().

Here is the call graph for this function:

6.37.2.4 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_FINALISE
 (TYPE(SOLVER_MATRICES_TYPE),pointer *SOLVER_MATRICES*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
 [private]

Finalises the solver matrices and deallocates all memory.

Parameters:

SOLVER_MATRICES A pointer to the solver matrices to finalise
ERR The error code
ERROR The error string

Definition at line 251 of file solver_matrices_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), KINDS::PTR, and SOLVER_MATRIX_FINALISE().

Referenced by SOLVER_MATRICES_CREATE_FINISH(), SOLVER_MATRICES_CREATE_START(), SOLVER_MATRICES_DESTROY(), and SOLVER_MATRICES_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.5 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE
 (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialises the solver matrices for a solver.

Parameters:

SOLVER A pointer to the problem solver to initialise the solver matrices for
ERR The error code
ERROR The error string

Definition at line 286 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `KINDS::PTR`, `SOLVER_MATRICES_FINALISE()`, and `SOLVER_MATRIX_INITIALISE()`.

Referenced by `SOLVER_MATRICES_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.6 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET` (`TYPE(SOLVER_MATRICES_TYPE)`,pointer *SOLVER_MATRICES*, `INTEGER(INTG),intent(in) LIBRARY_TYPE`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Sets the library type for the solver matrices (and vectors).

Parameters:

SOLVER_MATRICES A pointer to the solver matrices.

LIBRARY_TYPE The library type to set

See also:

`DISTRIBUTED_MATRIX_VECTOR_LibraryTypes`

ERR The error code

ERROR The error string

Definition at line 353 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.7 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT` (`INTEGER(INTG),intent(in) ID`, `TYPE(SOLVER_MATRICES_TYPE)`,pointer *SOLVER_MATRICES*, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR, *`)

Outputs the solver matrices.

Parameters:

ID The ID of the ouput stream

SOLVER_MATRICES A pointer to the solver matrices

ERR The error code

ERROR The error string

Definition at line 395 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `KINDS::PTR`.

Referenced by `SOLVER_ROUTINES::SOLVER_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.8 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET` (`TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES, INTEGER(INTG),dimension(:),intent(in) STORAGE_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`)

Sets the storage type (sparsity) of the solver matrices.

Parameters:

SOLVER_MATRICES A pointer to the solver matrices

STORAGE_TYPE `STORAGE_TYPE(matrix_idx)`. The storage type for the matrix_idx'th solver matrix

ERR The error code

ERROR The error string

Definition at line 444 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_DIAGONAL_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_ROW_COLUMN_STORAGE_TYPE`, `MATRIX_VECTOR::MATRIX_ROW_MAJOR_STORAGE_TYPE`, and `KINDS::PTR`.

Referenced by `SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH()`, and `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.37.2.9 subroutine `SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE` (`TYPE(SOLVER_MATRIX_TYPE),pointer SOLVER_MATRIX, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`)
[private]

Finalises a solver matrix and deallocates all memory.

Parameters:

SOLVER_MATRIX A pointer to the solver matrix to finalise

ERR The error code

ERROR The error string

Definition at line 781 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_MATRICES_FINALISE()`, and `SOLVER_MATRIX_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.37.2.10 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE
(TYPE(SOLVER_MATRICES_TYPE),pointer SOLVER_MATRICES,
INTEGER(INTG),intent(in) MATRIX_NUMBER, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialises a solver matrix.

Parameters:

SOLVER_MATRICES A pointer to the solver matrices to initialise
MATRIX_NUMBER The matrix number in the solver matrices to initialise
ERR The error code
ERROR The error string

Definition at line 809 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE`, `KINDS::PTR`, and `SOLVER_MATRIX_FINALISE()`.

Referenced by `SOLVER_MATRICES_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.37.2.11 subroutine SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_CALCULATE
(TYPE(SOLVER_MATRIX_TYPE),pointer SOLVER_MATRIX, INTEGER(INTG),intent(out) NUMBER_OF_
NON_ZEROS, INTEGER(INTG),dimension(:,),pointer ROW_INDICES,
INTEGER(INTG),dimension(:,),pointer COLUMN_INDICES,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*) [private]**

Calculates the structure (sparsity) of the solver matrix from the soluton mapping.

Parameters:

SOLVER_MATRIX A pointer to the solver matrix to calculate the structure for
NUMBER_OF_NON_ZEROS On return the number of non-zeros in the solver matrix
ROW_INDICES On return a pointer to row location indices in compressed row format. The pointers must be NULL on entry and the calling routine is responsible for deallocation.
COLUMN_INDICES On return a pointer to the column location indices in compressed row format. The pointers must be NULL on entry and the calling routine is responsible for deallocation.

ERR The error code

ERROR The error string

Definition at line 513 of file solver_matrices_routines.f90.

References `BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE,` `BASE_-ROUTINES::DIAGNOSTICS1,` `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`,
`BASE_ROUTINES::EXITS()`, `LISTS::LIST_CREATE_FINISH()`, `LISTS::LIST_CREATE_START()`,
`LISTS::LIST_DATA_TYPE_SET()`, `LISTS::LIST_DESTROY()`, `LISTS::LIST_INITIAL_SIZE_SET()`,
`LISTS::LIST_INTG_TYPE`, `LISTS::LIST_NUMBER_OF_ITEMS_GET()`, `LISTS::LIST_REMOVE_-DUPLICATES()`, and `KINDS::PTR`.

Referenced by `SOLVER_MATRICES_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38 SOLVER_ROUTINES Namespace Reference

This module handles all solver routines.

Functions

- subroutine [SOLVER_CREATE_FINISH](#) (SOLVER, ERR, ERROR,*)

Finishes the process of creating a solver for a problem solution.
- subroutine [SOLVER_CREATE_START](#) (SOLUTION, SOLVE_TYPE, SOLVER, ERR, ERROR,*)

Starts the process of creating a solver for a problem solution.
- subroutine [SOLVER_DESTROY](#) (SOLVER, ERR, ERROR,*)

Destroy a problem solver.
- subroutine [SOLVER_EIGENPROBLEM_CREATE_FINISH](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Finishes the process of creating a eigenproblem solver.
- subroutine [SOLVER_EIGENPROBLEM_FINALISE](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Finalise a eigenproblem solver for a problem solver.
- subroutine [SOLVER_EIGENPROBLEM_INITIALISE](#) (SOLVER, ERR, ERROR,*)

Initialise a eigenproblem solver for a problem solver.
- subroutine [SOLVER_EIGENPROBLEM_SOLVE](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Solve a eigenproblem solver.
- subroutine [SOLVER_FINALISE](#) (SOLVER, ERR, ERROR,*)

Finalises a problem solver and deallocates all memory.
- subroutine [SOLVER_INITIALISE](#) (SOLUTION, SOLVE_TYPE, ERR, ERROR,*)

Initialise a solver for a problem solution.
- subroutine [SOLVER_LIBRARY_SET](#) (SOLVER, SOLVER_LIBRARY, ERR, ERROR,*)

Sets/changes the type of library to use for the solver.
- subroutine [SOLVER_LINEAR_CREATE_FINISH](#) (LINEAR_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear solver.
- subroutine [SOLVER_LINEAR_DIRECT_CREATE_FINISH](#) (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear direct solver.
- subroutine [SOLVER_LINEAR_DIRECT_FINALISE](#) (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Finalise a direct linear solver for a linear solver and deallocate all memory.

- subroutine **SOLVER_LINEAR_DIRECT_INITIALISE** (LINEAR_SOLVER, ERR, ERROR,*)

Initialise a direct linear solver for a lienar solver.
- subroutine **SOLVER_LINEAR_DIRECT_SOLVE** (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Solve a linear direct solver.
- subroutine **SOLVER_LINEAR_DIRECT_TYPE_SET** (SOLVER, DIRECT_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of direct linear solver.
- subroutine **SOLVER_LINEAR_FINALISE** (LINEAR_SOLVER, ERR, ERROR,*)

Finalise a linear solver for a problem solver.
- subroutine **SOLVER_LINEAR_INITIALISE** (SOLVER, ERR, ERROR,*)

Initialise a linear solver for a problem solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET** (SOLVER, ABSOLUTE_TOLERANCE, ERR, ERROR,*)

Sets/changes the maximum absolute tolerance for an iterative linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_CREATE_FINISH** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear iterative solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET** (SOLVER, DIVERGENCE_TOLERANCE, ERR, ERROR,*)

Sets/changes the maximum divergence tolerance for an iterative linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_FINALISE** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Finalise an iterative linear solver for a linear solver and deallocate all memory.
- subroutine **SOLVER_LINEAR_ITERATIVE_INITIALISE** (LINEAR_SOLVER, ERR, ERROR,*)

Initialise an iterative linear solver for a linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET** (SOLVER, MAXIMUM_ITERATIONS, ERR, ERROR,*)

Sets/changes the maximum number of iterations for an iterative linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET** (SOLVER, ITERATIVE_PRECONDITIONER_TYPE, ERR, ERROR,*)

Sets/changes the type of preconditioner for an iterative linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET** (SOLVER, RELATIVE_TOLERANCE, ERR, ERROR,*)

Sets/changes the relative tolerance for an iterative linear solver.

- subroutine **SOLVER_LINEAR_ITERATIVE_SOLVE** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Solves a linear iterative linear solver.
- subroutine **SOLVER_LINEAR_ITERATIVE_TYPE_SET** (SOLVER, ITERATIVE_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of iterative linear solver.
- subroutine **SOLVER_LINEAR_SOLVE** (LINEAR_SOLVER, ERR, ERROR,*)

Solve a linear solver.
- subroutine **SOLVER_LINEAR_TYPE_SET** (SOLVER, LINEAR_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of linear solver.
- subroutine **SOLVER_MATRICES_ASSEMBLE** (SOLVER, ERR, ERROR,*)

Assembles the solver matrices and rhs from the equations.
- subroutine **SOLVER_NONLINEAR_CREATE_FINISH** (NONLINEAR_SOLVER, ERR, ERROR,*)

Finishes the process of creating a nonlinear solver.
- subroutine **SOLVER_NONLINEAR_FINALISE** (NONLINEAR_SOLVER, ERR, ERROR,*)

Finalise a nonlinear solver for a problem solver.
- subroutine **SOLVER_NONLINEAR_INITIALISE** (SOLVER, ERR, ERROR,*)

Initialise a nonlinear solver for a problem solver.
- subroutine **SOLVER_NONLINEAR_SOLVE** (NONLINEAR_SOLVER, ERR, ERROR,*)
 - subroutine **SOLVER_OUTPUT_TYPE_SET** (SOLVER, OUTPUT_TYPE, ERR, ERROR,*)

Sets/changes the output type for a solver.
- subroutine **SOLVER_SPARSITY_TYPE_SET** (SOLVER, SPARSITY_TYPE, ERR, ERROR,*)

Sets/changes the sparsity type for a solver.
- subroutine **SOLVER_TIME_INTEGRATION_CREATE_FINISH** (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Finishes the process of creating a time integration solver.
- subroutine **SOLVER_TIME_INTEGRATION_FINALISE** (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Finalise a time integration solver for a problem solver.
- subroutine **SOLVER_TIME_INTEGRATION_INITIALISE** (SOLVER, ERR, ERROR,*)

Initialise a time integration solver for a problem solver.
- subroutine **SOLVER_TIME_INTEGRATION_SOLVE** (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Solve a time integration solver.

- subroutine **SOLVER_SOLVE** (SOLVER, ERR, ERROR,*)
Solve the problem.
- subroutine **SOLVER_VARIABLES_UPDATE** (SOLVER, ERR, ERROR,*)
Updates the dependent variables from the solver solution.

Variables

- INTEGER(INTG), parameter **SOLVER_LINEAR_TYPE** = 1
Linear solution solver.
- INTEGER(INTG), parameter **SOLVER_NONLINEAR_TYPE** = 2
A nonlinear solution solver.
- INTEGER(INTG), parameter **SOLVER_TIME_INTEGRATION_TYPE** = 3
A time integration solver.
- INTEGER(INTG), parameter **SOLVER_EIGENPROBLEM_TYPE** = 4
A eigenproblem type.
- INTEGER(INTG), parameter **SOLVER_CMISS_LIBRARY** = LIBRARY_CMISS_TYPE
CMISS (internal) solver library.
- INTEGER(INTG), parameter **SOLVER_PETSC_LIBRARY** = LIBRARY_PETSC_TYPE
PETSc solver library.
- INTEGER(INTG), parameter **SOLVER_LINEAR_DIRECT_SOLVE_TYPE** = 1
Direct linear solver type.
- INTEGER(INTG), parameter **SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE** = 2
Iterative linear solver type.
- INTEGER(INTG), parameter **SOLVER_DIRECT_LU** = 1
LU direct linear solver.
- INTEGER(INTG), parameter **SOLVER_DIRECT_CHOLESKY** = 2
Cholesky direct linear solver.
- INTEGER(INTG), parameter **SOLVER_DIRECT_SVD** = 3
SVD direct linear solver.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_RICHARDSON** = 1
Richardson iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_CHEBYCHEV** = 2
Chebychev iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_CONJUGATE_GRADIENT** = 3

Conjugate gradient iterative solver type.

- INTEGER(INTG), parameter **SOLVER_ITERATIVE_BICONJUGATE_GRADIENT** = 4
Bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_GMRES** = 5
Generalised minimum residual iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_BICGSTAB** = 6
Stabalised bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_CONJGRAD_SQUARED** = 7
Conjugate gradient squared iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_NO_PRECONDITIONER** = 0
No preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_JACOBI_PRECONDITIONER** = 1
Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER** = 2
Iterative block Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_SOR_PRECONDITIONER** = 3
Successive over relaxation preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER** = 4
Incomplete Cholesky preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER** = 5
Incomplete LU preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER** = 6
Additive Schwrz preconditioner type.
- INTEGER(INTG), parameter **SOLVER_NO_OUTPUT** = 0
No output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_TIMING_OUTPUT** = 1
Timing output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_SOLVER_OUTPUT** = 2
Timing and solver specific output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_MATRIX_OUTPUT** = 3
Timing and solver specific output and solution matrices output from the solver routines.

- INTEGER(INTG), parameter **SOLVER_SPARSE_MATRICES** = 1
Use sparse solver matrices.
- INTEGER(INTG), parameter **SOLVER_FULL_MATRICES** = 2
Use fully populated solver matrices.
- INTEGER(INTG), parameter **SOLVER_EULER_INTEGRATOR** = 1
- INTEGER(INTG), parameter **SOLVER_IMPROVED_EULER_INTEGRATOR** = 2
- INTEGER(INTG), parameter **SOLVER_4TH_RUNGE_KUTTA_INTEGRATOR** = 3
- INTEGER(INTG), parameter **SOLVER_ADAMS_MOULTON_INTEGRATOR** = 4
- INTEGER(INTG), parameter **SOLVER_LSODA_INTEGRATOR** = 5

6.38.1 Detailed Description

This module handles all solver routines.

6.38.2 Function Documentation

6.38.2.1 subroutine **SOLVER_ROUTINES::SOLVER_CREATE_FINISH** `(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, *)`

Finishes the process of creating a solver for a problem solution.

Parameters:

SOLVER A pointer the solver to finish the creation of.

ERR The error code

ERROR The error string

Definition at line 202 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_EIGENPROBLEM_CREATE_FINISH()`, `SOLVER_EIGENPROBLEM_TYPE`, `SOLVER_FINALISE()`, `SOLVER_LINEAR_CREATE_FINISH()`, `SOLVER_LINEAR_TYPE`, `SOLVER_NONLINEAR_CREATE_FINISH()`, `SOLVER_NONLINEAR_TYPE`, `SOLVER_TIME_INTEGRATION_CREATE_FINISH()`, and `SOLVER_TIME_INTEGRATION_TYPE`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.2 subroutine **SOLVER_ROUTINES::SOLVER_CREATE_START** `(TYPE(SOLUTION_TYPE),pointer SOLUTION, INTEGER(INTG),intent(in)` `SOLVE_TYPE, TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out)` `ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Starts the process of creating a solver for a problem solution.

Parameters:

SOLUTION A pointer to the solution to create the solver for.

SOLVE_TYPE The type of solver to create

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

SOLVER On exit, a pointer to the problem solver.

ERR The error code

ERROR The error string

Definition at line 251 of file solver_routines.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::EXITS\(\)](#), [SOLVER_FINALISE\(\)](#), and [SOLVER_INITIALISE\(\)](#).

Referenced by [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.3 subroutine SOLVER_ROUTINES::SOLVER_DESTROY (TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Destroy a problem solver.

Parameters:

SOLVER A pointer the solver to destroy

ERR The error code

ERROR The error string

Definition at line 291 of file solver_routines.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::EXITS\(\)](#), and [SOLVER_FINALISE\(\)](#).

Referenced by [PROBLEM_ROUTINES::PROBLEM SOLUTION_FINALISE\(\)](#), and [PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.4 subroutine SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH (TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer EIGENPROBLEM_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finishes the process of creating a eigenproblem solver.

Parameters:

EIGENPROBLEM_SOLVER A pointer to the eigenproblem solver to finish the creation of.

ERR The error code

ERROR The error string

Definition at line 320 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.5 subroutine SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_FINALISE
`(TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer EIGENPROBLEM_SOLVER,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
`[private]`

Finalise a eigenproblem solver for a problem solver.

Parameters:

EIGENPROBLEM_SOLVER A pointer the eigenproblem solver to finalise

ERR The error code

ERROR The error string

Definition at line 349 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.6 subroutine SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialise a eigenproblem solver for a problem solver.

Parameters:

SOLVER A pointer the solver to initialise the eigenproblem solver for

ERR The error code

ERROR The error string

Definition at line 376 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.7 subroutine SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_SOLVE
 (TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer *EIGENPROBLEM_SOLVER*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
 [private]

Solve a eigenproblem solver.

Parameters:

EIGENPROBLEM_SOLVER A pointer the eigenproblem solver to solve

ERR The error code

ERROR The error string

Definition at line 412 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by SOLVER_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.8 subroutine SOLVER_ROUTINES::SOLVER_FINALISE (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Finalises a problem solver and deallocates all memory.

Parameters:

SOLVER A pointer the solver to finalise

ERR The error code

ERROR The error string

Definition at line 442 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), SOLVER_EIGENPROBLEM_FINALISE(), SOLVER_LINEAR_FINALISE(), SOLVER_NONLINEAR_FINALISE(), and SOLVER_TIME_INTEGRATION_FINALISE().

Referenced by SOLVER_CREATE_FINISH(), SOLVER_CREATE_START(), and SOLVER_DESTROY().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.9 subroutine SOLVER_ROUTINES::SOLVER_INITIALISE (TYPE(SOLUTION_TYPE),pointer *SOLUTION*, INTEGER(INTG),intent(in) *SOLVE_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Initialise a solver for a problem solution.

Parameters:

SOLUTION A pointer the solution to initialise the solver for

SOLVE_TYPE The type of solver to create

See also:

[SOLVER_ROUTINES::SolverTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 473 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_EIGENPROBLEM_INITIALISE()`, `SOLVER_EIGENPROBLEM_TYPE()`, `SOLVER_LINEAR_INITIALISE()`, `SOLVER_LINEAR_TYPE()`, `SOLVER_NO_OUTPUT()`, `SOLVER_NONLINEAR_INITIALISE()`, `SOLVER_NONLINEAR_TYPE()`, `SOLVER_SPARSE_MATRICES()`, `SOLVER_TIME_INTEGRATION_INITIALISE()`, and `SOLVER_TIME_INTEGRATION_TYPE()`.

Referenced by `SOLVER_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.10 subroutine `SOLVER_ROUTINES::SOLVER_LIBRARY_SET` (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(in) *SOLVER_LIBRARY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the type of library to use for the solver.

Parameters:

SOLVER A pointer the solver to set the type of

SOLVER_LIBRARY The type of library to use for the solver

See also:

[SOLVER_ROUTINES::SolverLibraries](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 545 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_CMISS_LIBRARY()`, `SOLVER_EIGENPROBLEM_TYPE()`, `SOLVER_LINEAR_DIRECT_SOLVE_TYPE()`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE()`, `SOLVER_LINEAR_TYPE()`, `SOLVER_NONLINEAR_TYPE()`, `SOLVER_PETSC_LIBRARY()`, and `SOLVER_TIME_INTEGRATION_TYPE()`.

Referenced by `LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.11 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH
 (**TYPE(LINEAR_SOLVER_TYPE)**,pointer **LINEAR_SOLVER**,
INTEGER(INTG),intent(out) ERR, **TYPE(VARYING_STRING),intent(out) ERROR**,
 *) [private]

Finishes the process of creating a linear solver.

Parameters:

LINEAR_SOLVER A pointer to the linear solver to finish the creation of.

ERR The error code

ERROR The error string

Definition at line 678 of file solver_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **SOLVER_LINEAR_DIRECT_CREATE_FINISH()**, **SOLVER_LINEAR_DIRECT_SOLVE_TYPE**, **SOLVER_LINEAR_ITERATIVE_CREATE_FINISH()**, and **SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE**.

Referenced by **SOLVER_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.12 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH
 (**TYPE(LINEAR_DIRECT_SOLVER_TYPE)**,pointer **LINEAR_DIRECT_SOLVER**,
INTEGER(INTG),intent(out) ERR, **TYPE(VARYING_STRING),intent(out) ERROR**,
 *) [private]

Finishes the process of creating a linear direct solver.

Parameters:

LINEAR_DIRECT_SOLVER A pointer to the linear direct solver to finish the creation of.

ERR The error code

ERROR The error string

Definition at line 717 of file solver_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, **SOLVER_CMISS_LIBRARY**, **SOLVER_FULL_MATRICES**, **SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH()**, **SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START()**, **SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET()**, **SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET()**, **SOLVER_PETSC_LIBRARY**, and **SOLVER_SPARSE_MATRICES**.

Referenced by **SOLVER_LINEAR_CREATE_FINISH()**.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.13 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_FINALISE
 (TYPE(LINEAR_DIRECT_SOLVER_TYPE),pointer *LINEAR_DIRECT_SOLVER*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Finalise a direct linear solver for a linear solver and deallocate all memory.

Parameters:

LINEAR_DIRECT_SOLVER A pointer to the linear direct solver to finalise

ERR The error code

ERROR The error string

Definition at line 784 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by SOLVER_LINEAR_FINALISE(), and SOLVER_LINEAR_TYPE_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.14 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE
 (TYPE(LINEAR_SOLVER_TYPE),pointer *LINEAR_SOLVER*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Initialise a direct linear solver for a linear solver.

Parameters:

LINEAR_SOLVER A pointer to the linear solver to initialise the direct solver for

ERR The error code

ERROR The error string

Definition at line 811 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), SOLVER_CMISS_LIBRARY, and SOLVER_DIRECT_LU.

Referenced by SOLVER_LINEAR_TYPE_SET().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.15 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE
 (TYPE(LINEAR_DIRECT_SOLVER_TYPE),pointer *LINEAR_DIRECT_SOLVER*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Solve a linear direct solver.

Parameters:

LINEAR_DIRECT_SOLVER A pointer to the linear direct solver to solve
ERR The error code
ERROR The error string

Definition at line 848 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `KINDS::PTR`, `SOLVER_CMISS_LIBRARY`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_LINEAR_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.16 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET
 (TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in)
 DIRECT_SOLVER_TYPE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *)**

Sets/changes the type of direct linear solver.

Parameters:

SOLVER A pointer the problem solver to set the direct linear solver type
DIRECT_SOLVER_TYPE The type of direct linear solver to set

See also:

[SOLVER_ROUTINES::DirectLinearSolverTypes](#),[SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 929 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_CMISS_LIBRARY`, `SOLVER_DIRECT_CHOLESKY`, `SOLVER_DIRECT_LU`, `SOLVER_DIRECT_SVD`, `SOLVER_LINEAR_DIRECT_SOLVE_TYPE`, `SOLVER_LINEAR_TYPE`, and `SOLVER_PETSC_LIBRARY`.

Here is the call graph for this function:

**6.38.2.17 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE (TYPE(LINEAR_SOLVER_TYPE),pointer LINEAR_SOLVER, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Finalise a linear solver for a problem solver.

Parameters:

LINEAR_SOLVER A pointer the linear solver to finalise
ERR The error code
ERROR The error string

Definition at line 1003 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_DIRECT_FINALISE()`, and `SOLVER_LINEAR_ITERATIVE_FINALISE()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.18 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_INITIALISE (`TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`)

Initialise a linear solver for a problem solver.

Parameters:

SOLVER A pointer the solver to initialise the linear solver for
ERR The error code
ERROR The error string

Definition at line 1032 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_ITERATIVE_INITIALISE()`, and `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`.

Referenced by `SOLVER_INITIALISE()`.

Here is the call graph for this function:

6.38.2.19 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ ABSOLUTE_TOLERANCE_SET (`TYPE(SOLVER_TYPE),pointer SOLVER,` `REAL(DP),intent(in) ABSOLUTE_TOLERANCE, INTEGER(INTG),intent(out) ERR,` `TYPE(VARYING_STRING),intent(out) ERROR, */`)

Sets/changes the maximum absolute tolerance for an iterative linear solver.

Parameters:

SOLVER A pointer the problem solver to set
ABSOLUTE_TOLERANCE The absolute tolerance to set
ERR The error code
ERROR The error string

Definition at line 1091 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, and `SOLVER_LINEAR_TYPE`.

Here is the call graph for this function:

6.38.2.20 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH (TYPE(LINEAR_ITERATIVE_SOLVER_TYPE),pointer LINEAR_ITERATIVE_SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finishes the process of creating a linear iterative solver.

Parameters:

LINEAR_ITERATIVE_SOLVER A pointer to the linear iterative solver to finish the creation of.
ERR The error code
ERROR The error string

Definition at line 1148 of file solver_routines.f90.

References COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), CMISS_PETSC::PETSC_KSPCREATE(), CMISS_PETSC::PETSC_KSPGETPC(), CMISS_PETSC::PETSC_KSPSETFROMOPTIONS(), CMISS_PETSC::PETSC_KSPSETOPERATORS(), CMISS_PETSC::PETSC_KSPSETTOLERANCES(), CMISS_PETSC::PETSC_KSPSETTYPE(), CMISS_PETSC::PCSETTYPE(), KINDS::PTR, SOLVER_CMISS_LIBRARY, SOLVER_FULL_MATRICES, SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER, SOLVER_ITERATIVE_BiCGSTAB, SOLVER_ITERATIVE_BICONJUGATE_GRADIENT, SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER, SOLVER_ITERATIVE_CHEBYCHEV, SOLVER_ITERATIVE_CONJGRAD_SQUARED, SOLVER_ITERATIVE_CONJUGATE_GRADIENT, SOLVER_ITERATIVE_GMRES, SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER, SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER, SOLVER_ITERATIVE_JACOBI_PRECONDITIONER, SOLVER_ITERATIVE_NO_PRECONDITIONER, SOLVER_ITERATIVE_RICHARDSON, SOLVER_ITERATIVE_SOR_PRECONDITIONER, SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET(), SOLVER_PETSC_LIBRARY, and SOLVER_SPARSE_MATRICES.

Referenced by SOLVER_LINEAR_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.21 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET (TYPE(SOLVER_TYPE),pointer SOLVER, REAL(DP),intent(in) DIVERGENCE_TOLERANCE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Sets/changes the maximum divergence tolerance for an iterative linear solver.

Parameters:

SOLVER A pointer the problem solver to set
DIVERGENCE_TOLERANCE The divergence tolerance to set
ERR The error code
ERROR The error string

Definition at line 1287 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, and `SOLVER_LINEAR_TYPE`.

Here is the call graph for this function:

6.38.2.22 subroutine `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE`
`(TYPE(LINEAR_ITERATIVE_SOLVER_TYPE),pointer LINEAR_ITERATIVE_SOLVER,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)`
`ERROR, *) [private]`

Finalise an iterative linear solver for a linear solver and deallocate all memory.

Parameters:

`LINEAR_ITERATIVE_SOLVER` A pointer the linear iterative solver to finalise

`ERR` The error code

`ERROR` The error string

Definition at line 1345 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `CMISS_PETSC::PETSC_KSPFINALISE()`, and `CMISS_PETSC::PETSC_PCFINALISE()`.

Referenced by `SOLVER_LINEAR_FINALISE()`, and `SOLVER_LINEAR_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.23 subroutine `SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE`
`(TYPE(LINEAR_SOLVER_TYPE),pointer LINEAR_SOLVER,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Initialise an iterative linear solver for a linear solver.

Parameters:

`LINEAR_SOLVER` A pointer the linear solver to initialise the iterative solver for

`ERR` The error code

`ERROR` The error string

Definition at line 1374 of file solver_routines.f90.

References `KINDS::_DP`, `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `CMISS_PETSC::PETSC_KSPINITIALISE()`, `CMISS_PETSC::PETSC_PCNINITIALISE()`, `SOLVER_ITERATIVE_GMRES`, `SOLVER_ITERATIVE_JACOBI_PRECONDITIONER`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_LINEAR_INITIALISE()`, and `SOLVER_LINEAR_TYPE_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.24 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_-
MAXIMUM_ITERATIONS_SET (TYPE(SOLVER_TYPE),pointer SOLVER,
INTEGER(INTG),intent(in) MAXIMUM_ITERATIONS, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Sets/changes the maximum number of iterations for an iterative linear solver.

Parameters:

SOLVER A pointer the problem solver to set the maximum number of iterations

MAXIMUM_ITERATIONS The maximum number of iterations

ERR The error code

ERROR The error string

Definition at line 1418 of file solver_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_-ROUTINES::EXITS()**, **SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE**, and **SOLVER_LINEAR_-TYPE**.

Here is the call graph for this function:

**6.38.2.25 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_-
PRECONDITIONER_TYPE_SET (TYPE(SOLVER_TYPE),pointer SOLVER,
INTEGER(INTG),intent(in) ITERATIVE_PRECONDITIONER_TYPE,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*)**

Sets/changes the type of preconditioner for an iterative linear solver.

Parameters:

SOLVER A pointer the problem solver to set the iterative linear solver type

ITERATIVE_PRECONDITIONER_TYPE The type of iterative preconditioner to set

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1475 of file solver_routines.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_-ROUTINES::EXITS()**, **SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER**, **SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER**, **SOLVER_ITERATIVE_-INCOMPLETE_CHOLESKY_PRECONDITIONER**, **SOLVER_ITERATIVE_INCOMPLETE_LU_-PRECONDITIONER**, **SOLVER_ITERATIVE_JACOBI_PRECONDITIONER**, **SOLVER_ITERATIVE_-NO_PRECONDITIONER**, **SOLVER_ITERATIVE_SOR_PRECONDITIONER**, **SOLVER_LINEAR_-ITERATIVE_SOLVE_TYPE**, **SOLVER_LINEAR_TYPE**, and **SOLVER_PETSC_LIBRARY**.

Here is the call graph for this function:

6.38.2.26 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET (TYPE(SOLVER_TYPE),pointer *SOLVER*,
REAL(DP),intent(in) *RELATIVE_TOLERANCE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the relative tolerance for an iterative linear solver.

Parameters:

SOLVER A pointer the solver to set
RELATIVE_TOLERANCE The relative tolerance to set
ERR The error code
ERROR The error string

Definition at line 1561 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE, and SOLVER_LINEAR_TYPE.

Here is the call graph for this function:

6.38.2.27 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE
 (TYPE(LINEAR_ITERATIVE_SOLVER_TYPE),pointer *LINEAR_ITERATIVE_SOLVER*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]

Solves a linear iterative linear solver.

Parameters:

LINEAR_ITERATIVE_SOLVER A pointer the linear iterative solver to solve
ERR The error code
ERROR The error string

Definition at line 1618 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), BASE_ROUTINES::GENERAL_OUTPUT_TYPE, CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON(), CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER(), CMISS_PETSC::PETSC_KSPGETRESIDUALNORM(), CMISS_PETSC::PETSC_KSPSOLVE(), KINDS::PTR, SOLVER_CMISS_LIBRARY, SOLVER_PETSC_LIBRARY, and SOLVER_SOLVER_OUTPUT.

Referenced by SOLVER_LINEAR_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.28 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET
 (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(in)
ITERATIVE_SOLVER_TYPE, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the type of iterative linear solver.

Parameters:

SOLVER A pointer the solver to set the iterative linear solver type

ITERATIVE_SOLVER_TYPE The type of iterative linear solver to set

See also:

[SOLVER_ROUTINES::IterativeLinearSolverTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1755 of file solver_routines.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::EXITS\(\)](#), [SOLVER_ITERATIVE_BICGSTAB](#), [SOLVER_ITERATIVE_BICONJUGATE_GRADIENT](#), [SOLVER_ITERATIVE_CHEBYCHEV](#), [SOLVER_ITERATIVE_CONJGRAD_SQUARED](#), [SOLVER_ITERATIVE_CONJUGATE_GRADIENT](#), [SOLVER_ITERATIVE_GMRES](#), [SOLVER_ITERATIVE_RICHARDSON](#), [SOLVER_LINEAR ITERATIVE SOLVE TYPE](#), [SOLVER_LINEAR TYPE](#), and [SOLVER_PETSC_LIBRARY](#).

Here is the call graph for this function:

6.38.2.29 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE (TYPE(LINEAR_Solver_TYPE),pointer LINEAR_Solver, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Solve a linear solver.

Parameters:

LINEAR_Solver A pointer to the linear solver to solve

ERR The error code

ERROR The error string

Definition at line 1836 of file solver_routines.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), [BASE_ROUTINES::EXITS\(\)](#), [SOLVER_LINEAR_DIRECT_SOLVE\(\)](#), [SOLVER_LINEAR_DIRECT_SOLVE_TYPE](#), [SOLVER_LINEAR ITERATIVE SOLVE\(\)](#), and [SOLVER_LINEAR ITERATIVE SOLVE_TYPE](#).

Referenced by [SOLVER_SOLVE\(\)](#).

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.30 subroutine SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET (TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) LINEAR_Solver_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Sets/changes the type of linear solver.

Parameters:

SOLVER A pointer the solver to set the linear solver type

LINEAR_SOLVER_TYPE The type of linear solver to set

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1875 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `SOLVER_LINEAR_DIRECT_FINALISE()`, `SOLVER_LINEAR_DIRECT_INITIALISE()`, `SOLVER_LINEAR_DIRECT_SOLVE_TYPE`, `SOLVER_LINEAR_ITERATIVE_FINALISE()`, `SOLVER_LINEAR_ITERATIVE_INITIALISE()`, `SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE`, and `SOLVER_LINEAR_TYPE`.

Here is the call graph for this function:

6.38.2.31 subroutine SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Assembles the solver matrices and rhs from the equations.

Parameters:

SOLVER A pointer to the solver

ERR The error code

ERROR The error string

Definition at line 1952 of file solver_routines.f90.

References `KINDS::DP`, `TIMER::CPU_TIMER()`, `BASE_ROUTINES::ENTERS()`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION`, `EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `FIELD_ROUTINES::FIELD_VALUES_SET_TYPE`, `BASE_ROUTINES::GENERAL_OUTPUT_TYPE`, `KINDS::PTR`, and `SOLVER_TIMING_OUTPUT`.

Referenced by `SOLVER_SOLVE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.32 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH
`(TYPE(NONLINEAR_SOLVER_TYPE),pointer NONLINEAR_SOLVER,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Finishes the process of creating a nonlinear solver.

Parameters:

NONLINEAR_SOLVER A pointer to the nonlinear solver to finish the creation of.

ERR The error code

ERROR The error string

Definition at line 2326 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.33 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE
`(TYPE(NONLINEAR_SOLVER_TYPE),pointer NONLINEAR_SOLVER,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*) [private]`

Finalise a nonlinear solver for a problem solver.

Parameters:

NONLINEAR_SOLVER A pointer the nonlinear solver to finalise

ERR The error code

ERROR The error string

Definition at line 2355 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `SOLVER_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.34 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE
`(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Initialise a nonlinear solver for a problem solver.

Parameters:

SOLVER A pointer the solver to initialise the nonlinear solver for

ERR The error code

ERROR The error string

Definition at line 2382 of file solver_routines.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `SOLVER_PETSC_LIBRARY`.

Referenced by `SOLVER_INITIALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.35 subroutine SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE
 (TYPE(NONLINEAR_SOLVER_TYPE),pointer *NONLINEAR_SOLVER*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
 *) [private]

Parameters:

NONLINEAR_SOLVER A pointer to the nonlinear solver to solve

ERR The error code

ERROR The error string

Definition at line 2438 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by SOLVER_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.36 subroutine SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET
 (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(in) *OUTPUT_TYPE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Sets/changes the output type for a solver.

Parameters:

SOLVER A pointer the problem solver to set the iterative linear solver type

OUTPUT_TYPE The type of solver output to be set

See also:

[SOLVER_ROUTINES::OutputTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 2467 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), SOLVER_MATRIX_OUTPUT, SOLVER_NO_OUTPUT, SOLVER_SOLVER_OUTPUT, and SOLVER_TIMING_OUTPUT.

Here is the call graph for this function:

6.38.2.37 subroutine SOLVER_ROUTINES::SOLVER_SOLVE (TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Solve the problem.

Parameters:

SOLVER A pointer the solver to solve

ERR The error code

ERROR The error string

Definition at line 2701 of file solver_routines.f90.

References TIMER::CPU_TIMER(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), BASE_ROUTINES::GENERAL_OUTPUT_TYPE, SOLVER_EIGENPROBLEM_SOLVE(), SOLVER_EIGENPROBLEM_TYPE, SOLVER_LINEAR_SOLVE(), SOLVER_LINEAR_TYPE, SOLVER_MATRICES_ASSEMBLE(), SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT(), SOLVER_MATRIX_OUTPUT, SOLVER_NONLINEAR_SOLVE(), SOLVER_NONLINEAR_TYPE, SOLVER_TIME_INTEGRATION_SOLVE(), SOLVER_TIME_INTEGRATION_TYPE, and SOLVER_TIMING_OUTPUT.

Referenced by PROBLEM_ROUTINES::PROBLEM SOLUTION_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.38 subroutine SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET

```
(TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(in) SPARSITY_TYPE,  
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)  
ERROR, *)
```

Sets/changes the sparsity type for a solver.

Parameters:

SOLVER A pointer the problem solver to set the iterative linear solver type

SPARSITY_TYPE The type of solver sparsity to be set

See also:

[SOLVER_ROUTINES::SparsityTypes](#), [SOLVER_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 2515 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_ROUTINES::EXITS(), SOLVER_FULL_MATRICES, and SOLVER_SPARSE_MATRICES.

Referenced by LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.2.39 subroutine SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_CREATE_FINISH (TYPE(TIME_INTEGRATION_SOLVER_TYPE),pointer **TIME_INTEGRATION_SOLVER**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Finishes the process of creating a time integration solver.

Parameters:

TIME_INTEGRATION_SOLVER A pointer to the time integration solver to finish the creation of.
ERR The error code
ERROR The error string

Definition at line 2560 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by SOLVER_CREATE_FINISH().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.40 subroutine SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_FINALISE
 (TYPE(TIME_INTEGRATION_SOLVER_TYPE),pointer TIME_INTEGRATION_-
 SOLVER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
 ERROR, *) [private]**

Finalise a time integration solver for a problem solver.

Parameters:

TIME_INTEGRATION_SOLVER A pointer the time integration solver to finalise
ERR The error code
ERROR The error string

Definition at line 2589 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by SOLVER_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.41 subroutine SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE
 (TYPE(SOLVER_TYPE),pointer SOLVER, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Initialise a time integration solver for a problem solver.

Parameters:

SOLVER A pointer the solver to initialise the time integration solver for
ERR The error code
ERROR The error string

Definition at line 2616 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS(), and SOLVER_PETSC_LIBRARY.

Referenced by SOLVER_INITIALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.42 subroutine SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE
(TYPE(TIME_INTEGRATION_SOLVER_TYPE),pointer *TIME_INTEGRATION_-
SOLVER*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *) [private]**

Solve a time integration solver.

Parameters:

TIME_INTEGRATION_SOLVER A pointer to the time integration solver to solve

ERR The error code

ERROR The error string

Definition at line 2672 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by SOLVER_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.38.2.43 subroutine SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE
(TYPE(SOLVER_TYPE),pointer *SOLVER*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Updates the dependent variables from the solver solution.

Parameters:

SOLVER A pointer the solver to update the variables from

ERR The error code

ERROR The error string

Definition at line 2778 of file solver_routines.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), FIELD_ROUTINES::FIELD_VALUES_SET_TYPE, and KINDS::PTR.

Referenced by PROBLEM_ROUTINES::PROBLEM SOLUTION_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.3 Variable Documentation

**6.38.3.1 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_4TH_RUNGE_-
KUTTA_INTEGRATOR = 3**

Definition at line 149 of file solver_routines.f90.

**6.38.3.2 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_ADAMS_MOULTON_-
INTEGRATOR = 4**

Definition at line 150 of file solver_routines.f90.

**6.38.3.3 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_EULER_-
INTEGRATOR = 1**

Definition at line 147 of file solver_routines.f90.

**6.38.3.4 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_IMPROVED_EULER_-
INTEGRATOR = 2**

Definition at line 148 of file solver_routines.f90.

**6.38.3.5 INTEGER(INTG),parameter SOLVER_ROUTINES::SOLVER_LSODA_-
INTEGRATOR = 5**

Definition at line 151 of file solver_routines.f90.

6.39 SORTING Namespace Reference

This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

Classes

- interface [BUBBLE_SORT](#)
- interface [HEAP_SORT](#)
- interface [SHELL_SORT](#)

Functions

- subroutine [BUBBLE_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [BUBBLE_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [BUBBLE_SORT_DP](#) (A, ERR, ERROR,*)
- subroutine [HEAP_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [HEAP_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [HEAP_SORT_DP](#) (A, ERR, ERROR,*)
- subroutine [SHELL_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [SHELL_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [SHELL_SORT_DP](#) (A, ERR, ERROR,*)

6.39.1 Detailed Description

This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

6.39.2 Function Documentation

6.39.2.1 subroutine [SORTING::BUBBLE_SORT_DP](#) (*REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, **) [private]

Definition at line 185 of file sorting.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.39.2.2 subroutine [SORTING::BUBBLE_SORT_INTG](#) (*INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */*) [private]

Definition at line 96 of file sorting.f90.

References [BASE_ROUTINES::ENTERS\(\)](#), [BASE_ROUTINES::ERRORS\(\)](#), and [BASE_ROUTINES::EXITS\(\)](#).

Here is the call graph for this function:

6.39.2.3 subroutine `SORTING::BUBBLE_SORT_SP` (`REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 140 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.39.2.4 subroutine `SORTING::HEAP_SORT_DP` (`REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 362 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.39.2.5 subroutine `SORTING::HEAP_SORT_INTG` (`INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 239 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.39.2.6 subroutine `SORTING::HEAP_SORT_SP` (`REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 300 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.39.2.7 subroutine `SORTING::SHELL_SORT_DP` (`REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 524 of file sorting.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.39.2.8 subroutine SORTING::SHELL_SORT_INTG (INTEGER(INTG),dimension(:),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 433 of file sorting.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.39.2.9 subroutine SORTING::SHELL_SORT_SP (REAL(SP),dimension(:),intent(inout) *A*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 478 of file sorting.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.40 STRINGS Namespace Reference

This module contains all string manipulation and transformation routines.

Classes

- interface [IS_ABBREVIATION](#)

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.

- interface [LIST_TO_CHARACTER](#)

Converts a list to its equivalent character string representation.

- interface [NUMBER_TO_CHARACTER](#)

Converts a number to its equivalent character string representation.

- interface [NUMBER_TO_VSTRING](#)

Converts a number to its equivalent varying string representation.

- interface [STRING_TO_DOUBLE](#)

Converts a string representation of a number to a double precision number.

- interface [STRING_TO_INTEGER](#)

Converts a string representation of a number to an integer.

- interface [STRING_TO_LONG_INTEGER](#)

Converts a string representation of a number to a long integer.

- interface [STRING_TO_LOGICAL](#)

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.

- interface [STRING_TO_SINGLE](#)

Converts a string representation of a number to a single precision number.

- interface [CHARACTER_TO_LOWERCASE](#)

Returns a character string which is the lowercase equivalent of the supplied string.

- interface [VSTRING_TO_LOWERCASE](#)

Returns a varying string which is the lowercase equivalent of the supplied string.

- interface [CHARACTER_TO_UPPERCASE](#)

Returns a character string which is the uppercase equivalent of the supplied string.

- interface [VSTRING_TO_UPPERCASE](#)

Returns a varying string which is the uppercase equivalent of the supplied string.

Functions

- LOGICAL [IS_ABBREVIATION_C_C](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
IS_ABBREVIATION returns .TRUE. if the character string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.
- LOGICAL [IS_ABBREVIATION_C_VS](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
IS_ABBREVIATION returns .TRUE. if the character string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.
- LOGICAL [IS_ABBREVIATION_VS_C](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
IS_ABBREVIATION returns .TRUE. if the varying string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.
- LOGICAL [IS_ABBREVIATION_VS_VS](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
IS_ABBREVIATION returns .TRUE. if the varying string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.
- LOGICAL [IS_DIGIT](#) (CHARAC)
IS_DIGIT returns .TRUE. if the character *CHARAC* is a digit character (i.e. 0..9).
- LOGICAL [IS_LETTER](#) (CHARAC)
IS_LETTER returns .TRUE. if the character *CHARAC* is a letter character (i.e. A..Z or a..z).
- LOGICAL [IS_LOWERCASE](#) (CHARC)
Returns .TRUE. *if the supplied character is a lowercase character.*
- LOGICAL [IS_UPPERCASE](#) (CHARC)
Returns .TRUE. *if the supplied character is an uppercase character.*
- LOGICAL [IS_WHITESPACE](#) (CHARAC)
IS_WHITESPACE returns .TRUE. if the character *CHARAC* is a whitespace character (i.e. space, tabs, etc.).
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_C](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR, LIST_LENGTHS)
Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_INTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
Converts an integer list to its equivalent character string representation as determined by the supplied format.
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_LINTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
Converts an long integer list to its equivalent character string representation as determined by the supplied format.
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_L](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.

- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_SP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a single precision list to its equivalent character string representation as determined by the supplied format string.

- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_DP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a double precision list to its equivalent character string representation as determined by the supplied format string.

- CHARACTER(LEN=MAXSTRLEN) [LOGICAL_TO_CHARACTER](#) (LOGICALVALUE, ERR, ERROR)

Converts a logical value to either a "TRUE" or "FALSE" character string.

- TYPE([VARYING_STRING](#)) [LOGICAL_TO_VSTRING](#) (LOGICALVALUE, ERR, ERROR)

Converts a logical value to either a "TRUE" or "FALSE" varying string.

- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_SP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_DP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE([VARYING_STRING](#)) [NUMBER_TO_VSTRING_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

- TYPE([VARYING_STRING](#)) [NUMBER_TO_VSTRING_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".

- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_SP (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_DP (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- REAL(DP) STRING_TO_DOUBLE_C (STRING, ERR, ERROR)

Converts a character string representation of a number to a double precision number.

- REAL(DP) STRING_TO_DOUBLE_VS (STRING, ERR, ERROR)

Converts a varying string representation of a number to a double precision number.

- INTEGER(INTG) STRING_TO_INTEGER_C (STRING, ERR, ERROR)

Converts a character string representation of a number to an integer.

- INTEGER(INTG) STRING_TO_INTEGER_VS (STRING, ERR, ERROR)

Converts a varying string representation of a number to an integer.

- INTEGER(LINTG) STRING_TO_LONG_INTEGER_C (STRING, ERR, ERROR)

Converts a character string representation of a number to a long integer.

- INTEGER(LINTG) STRING_TO_LONG_INTEGER_VS (STRING, ERR, ERROR)

Converts a varying string representation of a number to a long integer.

- LOGICAL STRING_TO_LOGICAL_C (STRING, ERR, ERROR)

Converts a character string representation of a boolean (TRUE or FALSE) to a logical.

- LOGICAL STRING_TO_LOGICAL_VS (STRING, ERR, ERROR)

Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.

- REAL(SP) STRING_TO_SINGLE_C (STRING, ERR, ERROR)

Converts a character string representation of a number to a single precision number.

- REAL(SP) STRING_TO_SINGLE_VS (STRING, ERR, ERROR)

Converts a varying string representation of a number to a single precision number.

- function CHARACTER_TO_LOWERCASE_C (STRING)

- function CHARACTER_TO_LOWERCASE_VS (STRING)

Returns a character string that is the lowercase equivalent of the supplied varying string.

- TYPE(VARYING_STRING) VSTRING_TO_LOWERCASE_C (STRING)

Returns a varying string that is the lowercase equivalent of the supplied character string.

- TYPE(VARYING_STRING) VSTRING_TO_LOWERCASE_VS (STRING)
Returns a varying string that is the lowercase equivalent of the supplied varying string.
- function CHARACTER_TO_UPPERCASE_C (STRING)
Returns a character string which is uppercase equivalent of the supplied character string.
- function CHARACTER_TO_UPPERCASE_VS (STRING)
Returns a character string which is uppercase equivalent of the supplied varying string.
- TYPE(VARYING_STRING) VSTRING_TO_UPPERCASE_C (STRING)
Returns a varying string which is uppercase equivalent of the supplied character string.
- TYPE(VARYING_STRING) VSTRING_TO_UPPERCASE_VS (STRING)
Returns a varying string which is uppercase equivalent of the supplied varying string.

6.40.1 Detailed Description

This module contains all string manipulation and transformation routines.

6.40.2 Function Documentation

6.40.2.1 function STRINGS::CHARACTER_TO_LOWERCASE_C (CHARACTER(LEN=*)**intent(in)** STRING) [private]

Parameters:

STRING The string to convert to lowercase

Definition at line 1609 of file strings.f90.

References IS_UPPERCASE().

Here is the call graph for this function:

6.40.2.2 function STRINGS::CHARACTER_TO_LOWERCASE_VS (TYPE(VARYING_STRING),**intent(in)** STRING) [private]

Returns a character string that is the lowercase equivalent of the supplied varying string.

Parameters:

STRING The string to convert

Definition at line 1634 of file strings.f90.

References IS_UPPERCASE().

Here is the call graph for this function:

**6.40.2.3 function STRINGS::CHARACTER_TO_UPPERCASE_C
(CHARACTER(LEN=*),intent(in) STRING) [private]**

Returns a character string which is uppercase equivalent of the supplied character string.

Parameters:

STRING The string to convert

Definition at line 1709 of file strings.f90.

References IS_LOWERCASE().

Here is the call graph for this function:

**6.40.2.4 function STRINGS::CHARACTER_TO_UPPERCASE_VS
(TYPE(VARYING_STRING),intent(in) STRING) [private]**

Returns a character string which is uppercase equivalent of the supplied varying string.

Parameters:

STRING The string to convert

Definition at line 1734 of file strings.f90.

References IS_LOWERCASE().

Here is the call graph for this function:

**6.40.2.5 LOGICAL STRINGS::IS_ABBREVIATION_C_C (CHARACTER(LEN=*),intent(in)
SHORT, CHARACTER(LEN=*),intent(in) LONG, INTEGER(INTG),intent(in)
MIN_NUM_CHARACTERS) [private]**

IS_ABBREVIATION returns .TRUE. if the character string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 160 of file strings.f90.

**6.40.2.6 LOGICAL STRINGS::IS_ABBREVIATION_C_VS (CHARACTER(LEN=*),intent(in)
SHORT, TYPE(VARYING_STRING),intent(in) LONG, INTEGER(INTG),intent(in)
MIN_NUM_CHARACTERS) [private]**

IS_ABBREVIATION returns .TRUE. if the character string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 192 of file strings.f90.

6.40.2.7 LOGICAL STRINGS::IS_ABBREVIATION_VS_C (TYPE(VARYING_STRING),intent(in) *SHORT*, CHARACTER(LEN=*),intent(in) *LONG*, INTEGER(INTG),intent(in) *MIN_NUM_CHARACTERS*) [private]

IS_ABBREVIATION returns .TRUE. if the varying string *SHORT* is an abbreviation of the character string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 224 of file strings.f90.

6.40.2.8 LOGICAL STRINGS::IS_ABBREVIATION_VS_VS (TYPE(VARYING_STRING),intent(in) *SHORT*, TYPE(VARYING_STRING),intent(in) *LONG*, INTEGER(INTG),intent(in) *MIN_NUM_CHARACTERS*) [private]

IS_ABBREVIATION returns .TRUE. if the varying string *SHORT* is an abbreviation of the varying string *LONG*. *SHORT* must be at least *MIN_NUM_CHARACTERS* long.

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 256 of file strings.f90.

6.40.2.9 LOGICAL STRINGS::IS_DIGIT (CHARACTER(LEN=1),intent(in) *CHARAC*)

IS_DIGIT returns .TRUE. if the character *CHARAC* is a digit character (i.e. 0..9).

Parameters:

CHARAC The character to test if it is a digit

Definition at line 287 of file strings.f90.

6.40.2.10 LOGICAL STRINGS::IS_LETTER (CHARACTER(LEN=1),intent(in) *CHARAC*)

IS_LETTER returns .TRUE. if the character *CHARAC* is a letter character (i.e. A..Z or a..z).

Parameters:

CHARAC The character to test if it is a letter

Definition at line 305 of file strings.f90.

6.40.2.11 LOGICAL STRINGS::IS_LOWERCase (CHARACTER(LEN=1),intent(in) *CHARC*) [private]

Returns .TRUE. if the supplied character is a lowercase character.

Parameters:

CHARC The character to test if it is lowercase

Definition at line 324 of file strings.f90.

Referenced by CHARACTER_TO_UPPERCASE_C(), CHARACTER_TO_UPPERCASE_VS(), VSTRING_TO_UPPERCASE_C(), and VSTRING_TO_UPPERCASE_VS().

Here is the caller graph for this function:

6.40.2.12 LOGICAL STRINGS::IS_UPPERCase (CHARACTER(LEN=1),intent(in) *CHARC*) [private]

Returns .TRUE. if the supplied character is an uppercase character.

Parameters:

CHARC The character to test if it is uppercase

Definition at line 346 of file strings.f90.

Referenced by CHARACTER_TO_LOWERCase_C(), CHARACTER_TO_LOWERCase_VS(), VSTRING_TO_LOWERCase_C(), and VSTRING_TO_LOWERCase_VS().

Here is the caller graph for this function:

6.40.2.13 LOGICAL STRINGS::IS_WHITESPACE (CHARACTER(LEN=1),intent(in) *CHARAC*)

IS_WHITESPACE returns .TRUE. if the character CHARAC is a whitespace character (i.e. space, tabs, etc.).

Parameters:

CHARAC The character to test if it is whitespace

Definition at line 368 of file strings.f90.

6.40.2.14 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_- CHARACTER_C (INTEGER(INTG),intent(in) *NUMBER_IN_LIST*, CHARACTER(LEN=*) ,dimension(*number_in_list*),intent(in) *LIST*, CHARACTER(LEN=*) ,intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, INTEGER(INTG),dimension(*number_in_list*),intent(in),optional *LIST_LENGTHS*) [private]

Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use. Ignored for character lists.

ERR The error code

ERROR The error string

LIST_LENGTHS LIST_LENGTHS(i). Optional, The length of the i'th list item.

Definition at line 387 of file strings.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, and **BASE_ROUTINES::EXITS()**.

Here is the call graph for this function:

6.40.2.15 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER_DP
(INTEGER(INTG),intent(in) NUMBER_IN_LIST, REAL(DP),dimension(number_in_list),intent(in) LIST, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]

Converts a double precision list to its equivalent character string representation as determined by the supplied format string.

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use for the conversion

ERR The error code

ERROR The error string

Definition at line 668 of file strings.f90.

References **BASE_ROUTINES::ENTERS()**, **BASE_ROUTINES::ERRORS()**, **BASE_ROUTINES::EXITS()**, and **NUMBER_TO_CHARACTER_DP()**.

Here is the call graph for this function:

6.40.2.16 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER_INTG
(INTEGER(INTG),intent(in) NUMBER_IN_LIST, INTEGER(INTG),dimension(number_in_list),intent(in) LIST, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]

Converts an integer list to its equivalent character string representation as determined by the supplied format.

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use for the conversion

ERR The error code

ERROR The error string

Definition at line 458 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_INTG()`.

Here is the call graph for this function:

**6.40.2.17 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER_L
(INTEGER(INTG),intent(in) NUMBER_IN_LIST, LOGICAL,dimension(number_in_list),intent(in) LIST, CHARACTER(LEN=*),intent(in) FORMAT,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST `LIST(i)`. The i'th item in the list

FORMAT The format to use. Ignored for logical lists.

ERR The error code

ERROR The error string

Definition at line 564 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `LOGICAL_TO_CHARACTER()`.

Here is the call graph for this function:

**6.40.2.18 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER_LINTG
(INTEGER(INTG),intent(in) NUMBER_IN_LIST,
INTEGER(LINTG),dimension(number_in_list),intent(in) LIST,
CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Converts an long integer list to its equivalent character string representation as determined by the supplied format.

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST `LIST(i)`. The i'th item in the list

FORMAT The format to use for the conversion

ERR The error code

ERROR The error string

Definition at line 511 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_LINTG()`.

Here is the call graph for this function:

**6.40.2.19 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER_SP
 (`INTEGER(INTG),intent(in) NUMBER_IN_LIST`, `REAL(SP),dimension(number_in_list)`,
`intent(in) LIST`, `CHARACTER(LEN=*)intent(in) FORMAT`,
`INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_STRING),intent(out) ERROR`)
[private]**

Converts a single precision list to its equivalent character string representation as determined by the supplied format string.

Parameters:

NUMBER_IN_LIST The number of items in the list
LIST `LIST(i)`. The i'th item in the list
FORMAT The format to use for the conversion
ERR The error code
ERROR The error string

Definition at line 616 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `NUMBER_TO_CHARACTER_SP()`.

Here is the call graph for this function:

**6.40.2.20 CHARACTER(LEN=MAXSTRLEN) STRINGS::LOGICAL_TO_CHARACTER
 (`LOGICAL,intent(in) LOGICALVALUE`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR`)**

Converts a logical value to either a "TRUE" or "FALSE" character string.

Parameters:

LOGICALVALUE The logical value to convert
ERR The error code
ERROR The error string

Definition at line 720 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `LIST_TO_CHARACTER_L()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.21 TYPE(VARYING_STRING) STRINGS::LOGICAL_TO_VSTRING
 (LOGICAL,intent(in) LOGICALVALUE, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Converts a logical value to either a "TRUE" or "FALSE" varying string.

Parameters:

- LOGICALVALUE** The logical value to convert
- ERR** The error code
- ERROR** The error string

Definition at line 750 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by INPUT_OUTPUT::WR(), INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L(), INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L(), and INPUT_OUTPUT::WRITE_STRING_VALUE_L().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.22 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER_DP
 (REAL(DP),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
 [private]**

Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

Parameters:

- NUMBER** The number to convert
- FORMAT** The format to use in the conversion
- ERR** The error code
- ERROR** The error string

Definition at line 947 of file strings.f90.

References KINDS::_DP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by LIST_TO_CHARACTER_DP().

Here is the call graph for this function:

Here is the caller graph for this function:

6.40.2.23 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER_INTG (INTEGER(INTG),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 780 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `LIST_TO_CHARACTER_INTG()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.40.2.24 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER_LINTG (INTEGER(LINTG),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 817 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `LIST_TO_CHARACTER_LINTG()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.25 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER_SP
(REAL(SP),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 854 of file strings.f90.

References KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by LIST_TO_CHARACTER_SP().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.40.2.26 TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING_DP
(REAL(DP),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1222 of file strings.f90.

References KINDS::_DP, and BASE_ROUTINES::ERRORS().

Here is the call graph for this function:

6.40.2.27 **TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING_INTG**
 (**INTEGER(INTG),intent(in)** *NUMBER*, **CHARACTER(LEN=*)**,**intent(in)** *FORMAT*,
INTEGER(INTG),intent(out) *ERR*, **TYPE(VARYING_STRING),intent(out)** *ERROR*)
 [private]

Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

Parameters:

NUMBER The number to convert
FORMAT The format to use in the conversion
ERR The error code
ERROR The error string

Definition at line 1039 of file strings.f90.

References BASE_ROUTINES::ERRORS().

Here is the call graph for this function:

6.40.2.28 **TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING_LINTG**
 (**INTEGER(LINTG),intent(in)** *NUMBER*, **CHARACTER(LEN=*)**,**intent(in)** *FORMAT*,
INTEGER(INTG),intent(out) *ERR*, **TYPE(VARYING_STRING),intent(out)** *ERROR*)
 [private]

Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".

Parameters:

NUMBER The number to convert
FORMAT The format to use in the conversion
ERR The error code
ERROR The error string

Definition at line 1082 of file strings.f90.

References BASE_ROUTINES::ERRORS().

Here is the call graph for this function:

6.40.2.29 **TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING_SP**
 (**REAL(SP),intent(in)** *NUMBER*, **CHARACTER(LEN=*)**,**intent(in)** *FORMAT*,
INTEGER(INTG),intent(out) *ERR*, **TYPE(VARYING_STRING),intent(out)** *ERROR*)
 [private]

Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1125 of file strings.f90.

References KINDS::_SP, BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.40.2.30 REAL(DP) STRINGS::STRING_TO_DOUBLE_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a character string representation of a number to a double precision number.

Parameters:

STRING The string to convert

ERR The error code !<The error code

ERROR The error string !<The error string

Definition at line 1322 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.40.2.31 REAL(DP) STRINGS::STRING_TO_DOUBLE_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a varying string representation of a number to a double precision number.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1349 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.40.2.32 INTEGER(INTG) STRINGS::STRING_TO_INTEGER_C
 (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Converts a character string representation of a number to an integer.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1380 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

**6.40.2.33 INTEGER(INTG) STRINGS::STRING_TO_INTEGER_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a number to an integer.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1407 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.40.2.34 LOGICAL STRINGS::STRING_TO_LOGICAL_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a character string representation of a boolean (TRUE or FALSE) to a logical.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1496 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Here is the call graph for this function:

6.40.2.35 LOGICAL STRINGS::STRING_TO_LOGICAL_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1523 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.40.2.36 INTEGER(LINTG) STRINGS::STRING_TO_LONG_INTEGER_C (CHARACTER(LEN=*) ,intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a character string representation of a number to a long integer.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1438 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

6.40.2.37 INTEGER(LINTG) STRINGS::STRING_TO_LONG_INTEGER_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Converts a varying string representation of a number to a long integer.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1465 of file strings.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.40.2.38 REAL(SP) STRINGS::STRING_TO_SINGLE_C (CHARACTER(LEN=*),intent(in)
STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR) [private]**

Converts a character string representation of a number to a single precision number.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1552 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.40.2.39 REAL(SP) STRINGS::STRING_TO_SINGLE_VS (TYPE(VARYING_-
STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Converts a varying string representation of a number to a single precision number.

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1579 of file strings.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.40.2.40 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_LOWERCASE_C
(CHARACTER(LEN=*),intent(in) STRING) [private]**

Returns a varying string that is the lowercase equivalent of the supplied character string.

Parameters:

STRING The string to convert

Definition at line 1659 of file strings.f90.

References IS_UPPERCASE().

Here is the call graph for this function:

**6.40.2.41 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_LOWERCASE_VS
(TYPE(VARYING_STRING),intent(in) STRING) [private]**

Returns a varying string that is the lowercase equivalent of the supplied varying string.

Parameters:

STRING The string to convert

Definition at line 1684 of file strings.f90.

References IS_UPPERCASE().

Here is the call graph for this function:

**6.40.2.42 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_UPPERCASE_C
(CHARACTER(LEN=*),intent(in) STRING) [private]**

Returns a varying string which is uppercase equivalent of the supplied character string.

Parameters:

STRING The string to convert

Definition at line 1759 of file strings.f90.

References IS_LOWERCASE().

Here is the call graph for this function:

**6.40.2.43 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_UPPERCASE_VS
(TYPE(VARYING_STRING),intent(in) STRING) [private]**

Returns a varying string which is uppercase equivalent of the supplied varying string.

Parameters:

STRING The string to convert

Definition at line 1784 of file strings.f90.

References IS_LOWERCASE().

Here is the call graph for this function:

6.41 TIMER Namespace Reference

This module contains routines for timing the program.

Classes

- interface [interface](#)

Functions

- subroutine [CPU_TIMER](#) (TIME_TYPE, TIME, ERR, ERROR,*)

Variables

- INTEGER(INTG), parameter [USER_CPU](#) = 1
- INTEGER(INTG), parameter [SYSTEM_CPU](#) = 2
- INTEGER(INTG), parameter [TOTAL_CPU](#) = 3

6.41.1 Detailed Description

This module contains routines for timing the program.

6.41.2 Function Documentation

6.41.2.1 subroutine [TIMER::CPU_TIMER](#) (INTEGER(INTG),intent(in) TIME_TYPE, REAL(SP),dimension(*),intent(out) TIME, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Definition at line 90 of file timer_f.f90.

References F90C::C2FSTRING(), BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_ROUTINES::EXITS().

Referenced by TIMER::interface::CPUTIMER(), EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM(), SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE(), and SOLVER_ROUTINES::SOLVER_SOLVE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.41.3 Variable Documentation

6.41.3.1 INTEGER(INTG),parameter [TIMER::SYSTEM_CPU](#) = 2

Definition at line 62 of file timer_f.f90.

6.41.3.2 INTEGER(INTG),parameter [TIMER::TOTAL_CPU](#) = 3

Definition at line 63 of file timer_f.f90.

6.41.3.3 INTEGER(INTG),parameter TIMER::USER_CPU = 1

Definition at line 61 of file timer_f.f90.

6.42 TREES Namespace Reference

Implements trees of base types.

Classes

- struct [TREE_NODE_TYPE](#)
- struct [TREE_TYPE](#)

Functions

- subroutine [TREE_CREATE_FINISH](#) (TREE, ERR, ERROR,*)
Finishes the creation of a tree created with TREE_CREATE_START.
- subroutine [TREE_CREATE_START](#) (TREE, ERR, ERROR,*)
Starts the creation of a tree and returns a pointer to the created tree.
- subroutine [TREE_DESTROY](#) (TREE, ERR, ERROR,*)
Destroys a tree.
- subroutine [TREE_DETACH_AND_DESTROY](#) (TREE, NUMBER_IN_TREE, TREE_VALUES, ERR, ERROR,*)
Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.
- subroutine [TREE_DETACH_IN_ORDER](#) (TREE, X, COUNT, TREE_VALUES, ERR, ERROR,*)
Detaches the tree values in order from the specified tree node and adds them to the tree values array.
- subroutine [TREE_FINALISE](#) (TREE, ERR, ERROR,*)
Finalises a tree and deallocates all memory.
- subroutine [TREE_INITIALISE](#) (TREE, ERR, ERROR,*)
Initialises a tree.
- subroutine [TREE_INSERT_TYPE_SET](#) (TREE, INSERT_TYPE, ERR, ERROR,*)
Sets/changes the insert type for a tree.
- subroutine [TREE_ITEM_DELETE](#) (TREE, KEY, ERR, ERROR,*)
Deletes a tree node specified by a key from a tree.
- subroutine [TREE_ITEM_INSERT](#) (TREE, KEY, VALUE, INSERT_STATUS, ERR, ERROR,*)
Inserts a tree node into a red-black tree.
- subroutine [TREE_NODE_FINALISE](#) (TREE, TREE_NODE, ERR, ERROR,*)
Finalises a tree node and deallocates all memory.
- subroutine [TREE_NODE_INITIALISE](#) (TREE, TREE_NODE, ERR, ERROR,*)
Initialises a tree node.

- subroutine **TREE_NODE_KEY_GET** (TREE, TREE_NODE, KEY, ERR, ERROR,*)

Gets the key at a specified tree node.
- subroutine **TREE_NODE_VALUE_GET** (TREE, TREE_NODE, VALUE, ERR, ERROR,*)

Gets the value at a specified tree node.
- subroutine **TREE_NODE_VALUE_SET** (TREE, TREE_NODE, VALUE, ERR, ERROR,*)

Sets the value at a specified tree node.
- subroutine **TREE_OUTPUT** (ID, TREE, ERR, ERROR,*)

Outputs a tree to the specified output stream ID.
- subroutine **TREE_OUTPUT_IN_ORDER** (ID, TREE, X, ERR, ERROR,*)

Outputs a tree in order to the specified output stream ID from the specified tree node.
- TYPE(**TREE_NODE_TYPE**), pointer **TREE_PREDECESSOR** (TREE, X, ERR, ERROR)

Returns the predecessor of a tree at a specified tree node.
- subroutine **TREE_SEARCH** (TREE, KEY, X, ERR, ERROR,*)

Searches a tree to see if it contains a key.
- TYPE(**TREE_NODE_TYPE**), pointer **TREE_SUCCESSOR** (TREE, X, ERR, ERROR)

Returns the successor of a tree at a specified tree node.

Variables

- INTEGER(INTG), parameter **TREE_BLACK_NODE** = 0

The black colour type for a tree node.
- INTEGER(INTG), parameter **TREE_RED_NODE** = 1

The red colour type for a tree node.
- INTEGER(INTG), parameter **TREE_NODE_INSERT_SUCESSFUL** = 1

Successful insert status.
- INTEGER(INTG), parameter **TREE_NODE_DUPLICATE_KEY** = 2

Duplicate key found for those trees that do not allow duplicate keys.
- INTEGER(INTG), parameter **TREE_DUPLICATES_ALLOWED_TYPE** = 1

Duplicate keys allowed tree type.
- INTEGER(INTG), parameter **TREE_NO_DUPLICATES_ALLOWED** = 2

No duplicate keys allowed tree type.

6.42.1 Detailed Description

Implements trees of base types.

6.42.2 Function Documentation

6.42.2.1 subroutine TREES::TREE_CREATE_FINISH (TYPE(TREE_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Finishes the creation of a tree created with TREE_CREATE_START.

See also:

[\(TREES::TREE_CREATE_START\).](#)

Parameters:

TREE A pointer to the tree to finish

ERR The error code

ERROR The error string.

Definition at line 123 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_- ROUTINES::EXITS(), TREE_FINALISE(), and TREE_NODE_INITIALISE().

Referenced by MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), and NODE_- ROUTINES::NODES_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.2 subroutine TREES::TREE_CREATE_START (TYPE(TREE_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Starts the creation of a tree and returns a pointer to the created tree.

See also:

[\(TREES::TREE_CREATE_FINISH\).](#)

Parameters:

TREE A pointer to the tree to create. Must not be associated on entry.

ERR The error code.

ERROR The error string.

Definition at line 167 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_- ROUTINES::EXITS(), TREE_DUPLICATES_ALLOWED_TYPE, TREE_FINALISE(), and TREE_- INITIALISE().

Referenced by MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), and NODE_- ROUTINES::NODES_CREATE_START().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.3 subroutine TREES::TREE_DESTROY (TYPE(TREE_TYPE),pointer *TREE*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Destroys a tree.

Parameters:

TREE A pointer to the tree to destroy

ERR The error code

ERROR The error string.

Definition at line 200 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), and TREE_FINALISE().

Referenced by MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE(), and NODE_-ROUTINES::NODES_FINALISE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.4 subroutine TREES::TREE_DETACH_AND_DESTROY (TYPE(TREE_-TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *NUMBER_IN_TREE*,
INTEGER(INTG),dimension(:),pointer *TREE_VALUES*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.

Parameters:

TREE A pointer to the tree to detach and destroy

NUMBER_IN_TREE On exit, the number in the array that has been detached

TREE_VALUES On exit, a pointer to the detached tree values. Must not be associated on entry.

ERR The error code

ERROR The error string.

Definition at line 228 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), BASE_-ROUTINES::EXITS(), TREE_DETACH_IN_ORDER(), and TREE_FINALISE().

Referenced by MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.5 subroutine TREES::TREE_DETACH_IN_ORDER (TYPE(TREE_TYPE),pointer
TREE, TYPE(TREE_NODE_TYPE),pointer *X*, INTEGER(INTG),intent(inout)
COUNT, INTEGER(INTG),dimension(:),intent(inout) *TREE_VALUES*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
[private]**

Detaches the tree values in order from the specified tree node and adds them to the tree values array.

Parameters:

TREE A pointer to the tree to detach
X A pointer to the specified tree node to detach from
COUNT The current number in the detached tree values array
TREE_VALUES The current detached tree values array
ERR The error code
ERROR The error string.

Definition at line 273 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_DETACH_AND_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.6 subroutine TREES::TREE_FINALISE (TYPE(TREE_TYPE),pointer TREE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
 [private]**

Finalises a tree and deallocates all memory.

Parameters:

TREE A pointer to the tree to finalise
ERR The error code
ERROR The error string.

Definition at line 317 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `TREE_NODE_FINALISE()`.

Referenced by `TREE_CREATE_FINISH()`, `TREE_CREATE_START()`, `TREE_DESTROY()`, and `TREE_DETACH_AND_DESTROY()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.7 subroutine TREES::TREE_INITIALISE (TYPE(TREE_TYPE),pointer TREE,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
 [private]**

Initialises a tree.

Parameters:

TREE A pointer to the tree to initialise
ERR The error code
ERROR The error string.

Definition at line 345 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.8 subroutine TREES::TREE_INSERT_TYPE_SET (TYPE(TREE_TYPE),pointer *TREE*, INTEGER(INTG),intent(in) *INSERT_TYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Sets/changes the insert type for a tree.

Parameters:

TREE A pointer to the tree

INSERT_TYPE The insert type to set

See also:

[TREES::TreeInsertTypes](#), [TREES::TreeInsertTypes](#)

ERR The error code

ERROR The error string.

Definition at line 377 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `TREE_DUPLICATES_ALLOWED_TYPE`, and `TREE_NO_DUPLICATES_ALLOWED`.

Referenced by `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE()`, and `NODE_ROUTINES::NODES_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.9 subroutine TREES::TREE_ITEM_DELETE (TYPE(TREE_TYPE),pointer *TREE*, INTEGER(INTG),intent(in) *KEY*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Deletes a tree node specified by a key from a tree.

Parameters:

TREE A pointer to the Red-Black tree to delete from

KEY A pointer to the tree node to delete

ERR The error code

ERROR The error string.

Definition at line 419 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `TREE_BLACK_NODE`, `TREE_RED_NODE`, and `TREE_SUCCESSOR()`.

Referenced by `NODE_ROUTINES::NODE_NUMBER_SET()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.10 subroutine TREES::TREE_ITEM_INSERT (TYPE(TREE_TYPE),pointer TREE, INTEGER(INTG),intent(in) KEY, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) INSERT_STATUS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Inserts a tree node into a red-black tree.

Parameters:

TREE A pointer to the Red-Black tree to insert into

KEY The key to insert

VALUE The value to insert

INSERT_STATUS On exit, the status of the insert

See also:

[TREES::TreeNodeInsertStatus](#), [TREES::TreeNodeInsertStatus](#)

ERR The error code

ERROR The error string.

Definition at line 668 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, `TREE_BLACK_NODE`, `TREE_NO_DUPLICATES_ALLOWED`, `TREE_NODE_DUPLICATE_KEY`, `TREE_NODE_INITIALISE()`, `TREE_NODE_INSERT_SUCESSFUL`, and `TREE_RED_NODE`.

Referenced by `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE()`, `NODE_ROUTINES::NODE_NUMBER_SET()`, and `NODE_ROUTINES::NODES_CREATE_START()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.11 subroutine TREES::TREE_NODE_FINALISE (TYPE(TREE_TYPE),pointer TREE, TYPE(TREE_NODE_TYPE),pointer TREE_NODE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Finalises a tree node and deallocates all memory.

Parameters:

TREE A pointer to the tree containing the tree node to finalise

TREE_NODE A pointer to the tree node to finalise

ERR The error code

ERROR The error string.

Definition at line 852 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `TREE_FINALISE()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.12 subroutine `TREES::TREE_NODE_INITIALISE` (`TYPE(TREE_TYPE),pointer TREE,`
`TYPE(TREE_NODE_TYPE),pointer TREE_NODE, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, */ [private]`**

Initialises a tree node.

Parameters:

`TREE` A pointer to the tree containing the tree node to initialise

`TREE_NODE` A pointer to the tree node to initialise

`ERR` The error code

`ERROR` The error string.

Definition at line 885 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_ROUTINES::EXITS()`, and `TREE_BLACK_NODE`.

Referenced by `TREE_CREATE_FINISH()`, and `TREE_ITEM_INSERT()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.13 subroutine `TREES::TREE_NODE_KEY_GET` (`TYPE(TREE_TYPE),pointer TREE,`
`TYPE(TREE_NODE_TYPE),pointer TREE_NODE, INTEGER(INTG),intent(out)`
`KEY, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)`
`ERROR, */`**

Gets the key at a specified tree node.

Parameters:

`TREE` A pointer to the tree containing the tree node

`TREE_NODE` A pointer to the tree node to get the key of

`KEY` On exit, the key of the specified tree node

`ERR` The error code

`ERROR` The error string.

Definition at line 923 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.42.2.14 subroutine TREES::TREE_NODE_VALUE_GET (TYPE(TREE_TYPE),pointer *TREE*,
TYPE(TREE_NODE_TYPE),pointer *TREE_NODE*, INTEGER(INTG),intent(out)
VALUE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Gets the value at a specified tree node.

Parameters:

TREE A pointer to the tree containing the tree node
TREE_NODE A pointer to the tree node to get the value of
VALUE On exit, the value of the specified tree node
ERR The error code
ERROR The error string.

Definition at line 961 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Referenced by NODE_ROUTINES::NODE_CHECK_EXISTS().

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.15 subroutine TREES::TREE_NODE_VALUE_SET (TYPE(TREE_TYPE),pointer *TREE*,
TYPE(TREE_NODE_TYPE),pointer *TREE_NODE*, INTEGER(INTG),intent(in)
VALUE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)**

Sets the value at a specified tree node.

Parameters:

TREE A pointer to the tree containing the tree node
TREE_NODE A pointer to the tree node to set the value of
VALUE The value of the specified tree node to set
ERR The error code
ERROR The error string.

Definition at line 999 of file trees.f90.

References BASE_ROUTINES::ENTERS(), BASE_ROUTINES::ERRORS(), and BASE_-ROUTINES::EXITS().

Here is the call graph for this function:

**6.42.2.16 subroutine TREES::TREE_OUTPUT (INTEGER(INTG),intent(in) *ID*,
TYPE(TREE_TYPE),pointer *TREE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)**

Outputs a tree to the specified output stream ID.

Parameters:

ID The ID of the output stream
TREE A pointer to the tree to search
ERR The error code
ERROR The error string.

Definition at line 1037 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, `BASE_-ROUTINES::EXITS()`, and `TREE_OUTPUT_IN_ORDER()`.

Referenced by `NODE_ROUTINES::NODES_CREATE_FINISH()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.17 subroutine TREES::TREE_OUTPUT_IN_ORDER (INTEGER(INTG),intent(in) ID, TYPE(TREE_TYPE),pointer TREE, TYPE(TREE_NODE_TYPE),pointer X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Outputs a tree in order to the specified output stream ID from the specified tree node.

Parameters:

ID The ID of the output stream
TREE A pointer to the tree to search
X A pointer to the tree node to output from
ERR The error code
ERROR The error string.

Definition at line 1073 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_-ROUTINES::EXITS()`.

Referenced by `TREE_OUTPUT()`.

Here is the call graph for this function:

Here is the caller graph for this function:

6.42.2.18 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_PREDECESSOR (TYPE(TREE_TYPE),pointer TREE, TYPE(TREE_NODE_TYPE),pointer X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Returns the predecessor of a tree at a specified tree node.

Parameters:

TREE A pointer to the Red-Black tree to find the predecessor of
X A pointer to the tree node to return the predecessor of

ERR The error code

ERROR The error string.

Definition at line 1128 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Here is the call graph for this function:

**6.42.2.19 subroutine TREES::TREE_SEARCH (TYPE(TREE_TYPE),pointer TREE,
INTEGER(INTG),intent(in) KEY, TYPE(TREE_NODE_TYPE),pointer X,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
*)**

Searches a tree to see if it contains a key.

Parameters:

TREE A pointer to the tree to search

KEY The key to search for

X On return a pointer to the tree node containing the key. If the key does not exist NULL is returned

ERR The error code

ERROR The error string.

Definition at line 1184 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by `NODE_ROUTINES::NODE_CHECK_EXISTS()`.

Here is the call graph for this function:

Here is the caller graph for this function:

**6.42.2.20 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_SUCCESSOR
(TYPE(TREE_TYPE),pointer TREE, TYPE(TREE_NODE_TYPE),pointer X,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Returns the successor of a tree at a specified tree node.

Parameters:

TREE A pointer to the Red-Black tree to find the successor of

X A pointer to the tree node to return the successor of

ERR The error code

ERROR The error string.

Definition at line 1241 of file trees.f90.

References `BASE_ROUTINES::ENTERS()`, `BASE_ROUTINES::ERRORS()`, and `BASE_ROUTINES::EXITS()`.

Referenced by TREE_ITEM_DELETE().

Here is the call graph for this function:

Here is the caller graph for this function:

6.43 TYPES Namespace Reference

This module contains all type definitions in order to avoid cyclic module references.

Classes

- struct [QUADRATURE_SCHEME_TYPE](#)
Contains information for a particular quadrature scheme.
- struct [QUADRATURE_SCHEME_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [QUADRATURE_SCHEME_TYPE](#).
- struct [QUADRATURE_TYPE](#)
Contains information on the quadrature to be used for integrating a basis.
- struct [BASIS_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [BASIS_TYPE](#).
- struct [BASIS_TYPE](#)
Contains all information about a basis .
- struct [BASIS_FUNCTIONS_TYPE](#)
Contains information on the defined basis functions.
- struct [COORDINATE_SYSTEM_TYPE](#)
Contains information on a coordinate system.
- struct [NODE_TYPE](#)
Contains information about a node.
- struct [NODES_TYPE](#)
Contains information on the nodes defined on a region.
- struct [MESH_DOFS_TYPE](#)
Contains information on the dofs for a mesh.
- struct [MESH_ELEMENT_TYPE](#)
Contains the information for an element in a mesh.
- struct [MESH_ELEMENTS_TYPE](#)
Contains the information for the elements of a mesh.
- struct [MESH_NODE_TYPE](#)
Contains the topology information for a global node of a mesh.
- struct [MESH_NODES_TYPE](#)
Contains the information for the nodes of a mesh.
- struct [MESH_TOPOLOGY_TYPE](#)

Contains information on the (global) topology of a mesh.

- struct [MESH_TOPOLOGY_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [MESH_TOPOLOGY_TYPE](#).

- struct [MESH_TYPE](#)

Contains information on a mesh defined on a region.

- struct [MESH_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [MESH_TYPE](#).

- struct [MESHES_TYPE](#)

Contains information on the meshes defined on a region.

- struct [GENERATED_MESH_REGULAR_TYPE](#)

Contains information on a generated regular mesh.

- struct [GENERATED_MESH_TYPE](#)

Contains information on a generated mesh.

- struct [DOMAIN_DOFS_TYPE](#)

Contains information on the degrees-of-freedom (dofs) for a domain.

- struct [DOMAIN_LINE_TYPE](#)

Contains the information for a line in a domain.

- struct [DOMAIN_LINE_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [DOMAIN_LINE_TYPE](#).

- struct [DOMAIN_LINES_TYPE](#)

Contains the topology information for the lines of a domain.

- struct [DOMAIN_FACE_TYPE](#)

Contains the information for a face in a domain.

- struct [DOMAIN_FACE_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).

- struct [DOMAIN_FACES_TYPE](#)

Contains the topology information for the faces of a domain.

- struct [DOMAIN_ELEMENT_TYPE](#)

Contains the information for an element in a domain.

- struct [DOMAIN_ELEMENTS_TYPE](#)

Contains the topology information for the elements of a domain.

- struct [DOMAIN_NODE_TYPE](#)

Contains the topology information for a local node of a domain.

- struct [DOMAIN_NODES_TYPE](#)
Contains the topology information for the nodes of a domain.
- struct [DOMAIN_TOPOLOGY_TYPE](#)
Contains the topology information for a domain.
- struct [DISTRIBUTED_VECTOR_TRANSFER_TYPE](#)
Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.
- struct [DISTRIBUTED_VECTOR_CMISS_TYPE](#)
Contains information for a CMISS distributed vector.
- struct [DISTRIBUTED_VECTOR_PETSC_TYPE](#)
Contains information for a PETSc distributed vector.
- struct [DISTRIBUTED_VECTOR_TYPE](#)
Contains the information for a vector that is distributed across a number of domains.
- struct [DISTRIBUTED_MATRIX_CMISS_TYPE](#)
Contains information for a CMISS distributed matrix.
- struct [DISTRIBUTED_MATRIX_PETSC_TYPE](#)
Contains information for a PETSc distributed matrix.
- struct [DISTRIBUTED_MATRIX_TYPE](#)
Contains the information for a matrix that is distributed across a number of domains.
- struct [VECTOR_TYPE](#)
Contains information for a vector.
- struct [MATRIX_TYPE](#)
Contains information for a matrix.
- struct [DOMAIN_ADJACENT_DOMAIN_TYPE](#)
Contains the information on an adjacent domain to a domain in a domain mapping.
- struct [DOMAIN_GLOBAL_MAPPING_TYPE](#)
Contains the local information for a global mapping number for a domain mapping.
- struct [DOMAIN_MAPPING_TYPE](#)
Contains information on the domain mappings (i.e., local and global numberings).
- struct [DOMAIN_MAPPINGS_TYPE](#)
Contains information on the domain decomposition mappings.
- struct [DOMAIN_TYPE](#)
A pointer to the domain decomposition for this domain.
- struct [DOMAIN_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [DOMAIN_TYPE](#).

- struct [DECOMPOSITION_LINE_TYPE](#)
Contains the information for a line in a decomposition.
- struct [DECOMPOSITION_LINES_TYPE](#)
Contains the topology information for the lines of a decomposition.
- struct [DECOMPOSITION_ELEMENT_TYPE](#)
Contains the information for an element in a decomposition.
- struct [DECOMPOSITION_ELEMENTS_TYPE](#)
Contains the topology information for the elements of a decomposition.
- struct [DECOMPOSITION_TOPOLOGY_TYPE](#)
Contains the topology information for a decomposition.
- struct [DECOMPOSITION_TYPE](#)
Contains information on the domain decomposition.
- struct [DECOMPOSITION_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [DECOMPOSITION_TYPE](#).
- struct [DECOMPOSITIONS_TYPE](#)
Contains information on the domain decompositions defined on a mesh.
- struct [FIELD_INTERPOLATED_POINT_METRICS_TYPE](#)
Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.
- struct [FIELD_INTERPOLATED_POINT_TYPE](#)
Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old CMISS name XG.
- struct [FIELD_INTERPOLATION_PARAMETERS_TYPE](#)
Contains the parameters required to interpolate a field variable within an element. Old CMISS name XE.
- struct [FIELD_GEOMETRIC_PARAMETERS_TYPE](#)
Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.
- struct [FIELD_SCALING_TYPE](#)
A type to hold the scale factors for the appropriate mesh component of a field.
- struct [FIELD_SCALINGS_TYPE](#)
A type to hold the field scalings for the field.
- struct [FIELD_DOF_TO_PARAM_MAP_TYPE](#)
A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).
- struct [FIELD_PARAM_TO_DOF_MAP_TYPE](#)
A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

- struct [FIELD_VARIABLE_COMPONENT_TYPE](#)
Contains information for a component of a field variable.
- struct [FIELD_VARIABLE_TYPE](#)
Contains information for a field variable defined on a field.
- struct [FIELD_VARIABLE_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).
- struct [FIELD_CREATE_VALUES_CACHE_TYPE](#)
A type to temporarily hold (cache) the user modifiable values which are used to create a field.
- struct [FIELD_MAPPINGS_TYPE](#)
The type containing the mappings for the field.
- struct [FIELD_PARAMETER_SET_TYPE](#)
A type to hold the parameter sets for a field.
- struct [FIELD_PARAMETER_SET_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_PARAMETER_SET_TYPE](#).
- struct [FIELD_PARAMETER_SETS_TYPE](#)
A type to store the parameter sets for a field.
- struct [FIELD_TYPE](#)
Contains information for a field defined on a region.
- struct [FIELD_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_TYPE](#).
- struct [FIELDS_TYPE](#)
Contains information on the fields defined on a region.
- struct [ELEMENT_MATRIX_TYPE](#)
Contains information for an element matrix.
- struct [ELEMENT_VECTOR_TYPE](#)
Contains information for an element vector.
- struct [EQUATIONS_MATRIX_TYPE](#)
Contains information about an equations matrix.
- struct [EQUATIONS_MATRIX_PTR_TYPE](#)
- struct [EQUATIONS_MATRICES_TYPE](#)
Contains information on the equations matrices and rhs vector.
- struct [VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE](#)
Contains the information about the mapping of a variable DOF to an equations matrix column.

- struct [VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE](#)
Contains the mapping for a dependent variable type to the equations matrices.
- struct [SOURCE_EQUATIONS_MATRICES_MAP_TYPE](#)
- struct [EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE](#)
- struct [EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE](#)
- struct [EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE](#)
- struct [EQUATIONS_MAPPING_TYPE](#)
- struct [EQUATIONS_INTERPOLATION_TYPE](#)
Contains information on the interpolation for the equations.
- struct [EQUATIONS_LINEAR_DATA_TYPE](#)
Contains information on any data required for a linear solution.
- struct [EQUATIONS_NONLINEAR_DATA_TYPE](#)
Contains information on any data required for a non-linear solution.
- struct [EQUATIONS_TIME_DATA_TYPE](#)
Contains information on any data required for a time-dependent equations.
- struct [EQUATIONS_TYPE](#)
Contains information about the equations in an equations set.
- struct [EQUATIONS_SET_GEOMETRY_TYPE](#)
Contains information on the geometry for an equations set.
- struct [EQUATIONS_SET_MATERIALS_TYPE](#)
- struct [EQUATIONS_SET_FIXED_CONDITIONS_TYPE](#)
Contains information on the fixed conditions for the problem.
- struct [EQUATIONS_SET_DEPENDENT_TYPE](#)
Contains information on the dependent variables for the equations set.
- struct [EQUATIONS_SET_SOURCE_TYPE](#)
Contains information on the source for the equations set.
- struct [EQUATIONS_SET_ANALYTIC_TYPE](#)
Contains information on the analytic setup for the equations set.
- struct [EQUATIONS_SET_TYPE](#)
Contains information on an equations set.
- struct [EQUATIONS_SET_PTR_TYPE](#)
- struct [EQUATIONS_SETS_TYPE](#)
- struct [SOLVER_MATRIX_TYPE](#)
Contains information on the solver matrix.
- struct [SOLVER_MATRIX_PTR_TYPE](#)
- struct [SOLVER_MATRICES_TYPE](#)
Contains information on the solver matrices and rhs vector.

- struct [LINEAR_DIRECT_SOLVER_TYPE](#)
Contains information for a direct linear solver.
- struct [LINEAR_ITERATIVE_SOLVER_TYPE](#)
Contains information for a direct linear solver.
- struct [LINEAR_SOLVER_TYPE](#)
Contains information for a linear solver.
- struct [NONLINEAR_SOLVER_TYPE](#)
Contains information for a nonlinear solver.
- struct [TIME_INTEGRATION_SOLVER_TYPE](#)
Contains information for a time integration solver.
- struct [EIGENPROBLEM_SOLVER_TYPE](#)
Contains information for a time integration solver.
- struct [SOLVER_TYPE](#)
Contains information on the type of solver to be used.
- struct [EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE](#)
- struct [EQUATIONS_TO_SOLVER_MAPS_TYPE](#)
- struct [EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [EQUATIONS_TO_SOLVER_MAPS_TYPE](#).
- struct [VARIABLE_TO_SOLVER_COL_MAP_TYPE](#)
Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).
- struct [EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE](#)
Contains information on the equations to solver matrix mappings when indexing by solver matrix number.
- struct [EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE](#)
Contains information on the equations to solver matrix mappings when indexing by equations matrix number.
- struct [EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE](#)
Contains information on the mapping from the equations rows in an equations set to the solver rows.
- struct [EQUATIONS_SET_TO_SOLVER_MAP_TYPE](#)
Contains information on the mappings from an equations set to the solver matrices.
- struct [SOLVER_COL_TO_EQUATIONS_MAP_TYPE](#)
Contains information about the mapping from a solver matrix column to equations matrices and variables.
- struct [SOLVER_COL_TO_VARIABLE_MAP_TYPE](#)
Contains information about mapping the solver column (solver dof) to the field variable dofs in the equations set.

- struct [SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE](#)
Contains information about the mappings from a solver matrix to the equations in an equations set.
- struct [SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE](#)
Contains information on the mappings from a solver matrix to equations sets.
- struct [SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE](#)
Contains information on the mappings from a solver row to the equations rows.
- struct [SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE](#)
Contains information about the cached create values for a solution mapping.
- struct [SOLUTION_MAPPING_TYPE](#)
Contains information on the solution mapping between the global equation sets and the solver matrices.
- struct [SOLUTION_TYPE](#)
Contains information on the solution of a problem.
- struct [SOLUTION_PTR_TYPE](#)
- struct [PROBLEM_LINEAR_DATA_TYPE](#)
Contains information on any data required for a linear solution.
- struct [PROBLEM_NONLINEAR_DATA_TYPE](#)
Contains information on any data required for a non-linear solution.
- struct [PROBLEM_TIME_DATA_TYPE](#)
Contains information on any data required for a time-dependent solution.
- struct [PROBLEM_CONTROL_TYPE](#)
- struct [PROBLEM_TYPE](#)
Contains information for a problem.
- struct [PROBLEM_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [PROBLEM_TYPE](#).
- struct [PROBLEMS_TYPE](#)
Contains information on the problems defined on a region.
- struct [REGION_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [REGION_TYPE](#).
- struct [REGION_TYPE](#)
Contains information for a region.

6.43.1 Detailed Description

This module contains all type definitions in order to avoid cyclic module references.

Chapter 7

Class Documentation

7.1 BASE_ROUTINES::FLAG_ERROR Interface Reference

Flags an error condition.

Private Member Functions

- subroutine [FLAG_ERROR_C](#) (STRING, ERR, ERROR,*)
- subroutine [FLAG_ERROR_VS](#) (STRING, ERR, ERROR,*)

7.1.1 Detailed Description

Flags an error condition.

See also:

[BASE_ROUTINES](#)

Definition at line 190 of file base_routines.f90.

7.1.2 Member Function Documentation

7.1.2.1 subroutine BASE_ROUTINES::FLAG_ERROR::FLAG_ERROR_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 470 of file base_routines.f90.

**7.1.2.2 subroutine BASE_ROUTINES::FLAG_ERROR::FLAG_ERROR_VS
(TYPE(VARYING_STRING),intent(in) *STRING*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

STRING The error condition string

ERR The error code

ERROR The error string

Definition at line 491 of file base_routines.f90.

7.2 BASE_ROUTINES::FLAG_WARNING Interface Reference

Flags a warning to the user.

Private Member Functions

- subroutine [FLAG_WARNING_C](#) (STRING, ERR, ERROR,*)
- subroutine [FLAG_WARNING_VS](#) (STRING, ERR, ERROR,*)

7.2.1 Detailed Description

Flags a warning to the user.

See also:

[BASE_ROUTINES](#)

Definition at line 196 of file base_routines.f90.

7.2.2 Member Function Documentation

**7.2.2.1 subroutine BASE_ROUTINES::FLAG_WARNING::FLAG_WARNING_C
(CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 510 of file base_routines.f90.

**7.2.2.2 subroutine BASE_ROUTINES::FLAG_WARNING::FLAG_WARNING_VS
(TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]**

Parameters:

STRING The warning string

ERR The error code

ERROR The error string

Definition at line 531 of file base_routines.f90.

7.3 BASE_ROUTINES::interface Interface Reference

Private Member Functions

- subroutine **CPUTIMER** (RETURN_TIME, TIME_TYPE, ERR, CERROR)

7.3.1 Detailed Description

Definition at line 173 of file base_routines.f90.

7.3.2 Member Function Documentation

7.3.2.1 subroutine BASE_ROUTINES::interface::CPUTIMER (REAL(DP),intent(out) RETURN_TIME, INTEGER(INTG),intent(in) TIME_TYPE, INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(*),intent(out) CERROR) [private]

Definition at line 178 of file base_routines.f90.

7.4 BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE Struct Reference

Contains information for an item in the routine list for diagnostics or timing.

Collaboration diagram for BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE:

Private Attributes

- TYPE(VARYING_STRING) NAME

Name of the routine.

- INTEGER(INTG) NUMBER_OF_INVOCATIONS

Number of times the routine has been invoked.

- REAL(SP) TOTAL_INCLUSIVE_CPU_TIME

Total User CPU time spent in the routine inclusive of calls.

- REAL(SP) TOTAL_INCLUSIVE_SYSTEM_TIME

Total System CPU time spent in the routine inclusive of calls.

- REAL(SP) TOTAL_EXCLUSIVE_CPU_TIME

Total User CPU time spent in the routine exclusive of calls.

- REAL(SP) TOTAL_EXCLUSIVE_SYSTEM_TIME

Total System CPU time spent in the routine exclusive of calls.

- TYPE(ROUTINE_LIST_ITEM_TYPE), pointer NEXT_ROUTINE

Pointer to the next routine item in the routine list.

7.4.1 Detailed Description

Contains information for an item in the routine list for diagnostics or timing.

Definition at line 110 of file base_routines.f90.

7.4.2 Member Data Documentation

7.4.2.1 TYPE(VARYING_STRING) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::NAME [private]

Name of the routine.

Definition at line 111 of file base_routines.f90.

**7.4.2.2 `TYPE(ROUTINE_LIST_ITEM_TYPE),pointer BASE_ROUTINES::ROUTINE_LIST_-
ITEM_TYPE::NEXT_ROUTINE [private]`**

Pointer to the next routine item in the routine list.

Definition at line 117 of file base_routines.f90.

**7.4.2.3 `INTEGER(INTG) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::NUMBER_OF_-
INVOCATIONS [private]`**

Number of times the routine has been invoked.

Definition at line 112 of file base_routines.f90.

**7.4.2.4 `REAL(SP) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::TOTAL_-
EXCLUSIVE_CPU_TIME [private]`**

Total User CPU time spent in the routine exclusive of calls.

Definition at line 115 of file base_routines.f90.

**7.4.2.5 `REAL(SP) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::TOTAL_-
EXCLUSIVE_SYSTEM_TIME [private]`**

Total System CPU time spent in the routine exclusive of calls.

Definition at line 116 of file base_routines.f90.

**7.4.2.6 `REAL(SP) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::TOTAL_INCLUSIVE_-
CPU_TIME [private]`**

Total User CPU time spent in the routine inclusive of calls.

Definition at line 113 of file base_routines.f90.

**7.4.2.7 `REAL(SP) BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE::TOTAL_INCLUSIVE_-
SYSTEM_TIME [private]`**

Total System CPU time spent in the routine inclusive of calls.

Definition at line 114 of file base_routines.f90.

7.5 BASE_ROUTINES::ROUTINE_LIST_TYPE Struct Reference

Contains information for the routine list for diagnostics or timing.

Collaboration diagram for BASE_ROUTINES::ROUTINE_LIST_TYPE:

Private Attributes

- TYPE(ROUTINE_LIST_ITEM_TYPE), pointer HEAD
A pointer to the head of the routine list.

7.5.1 Detailed Description

Contains information for the routine list for diagnostics or timing.

Definition at line 121 of file base_routines.f90.

7.5.2 Member Data Documentation

7.5.2.1 TYPE(ROUTINE_LIST_ITEM_TYPE),pointer BASE_ROUTINES::ROUTINE_LIST_TYPE::HEAD [private]

A pointer to the head of the routine list.

Definition at line 122 of file base_routines.f90.

7.6 BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE Struct Reference

Contains information for an item in the routine invocation stack.

Collaboration diagram for BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE:

Private Attributes

- TYPE(VARYING_STRING) NAME
The name of the routine.
- REAL(SP) INCLUSIVE_CPU_TIME
User CPU time spent in the routine inclusive of calls.
- REAL(SP) INCLUSIVE_SYSTEM_TIME
System CPU time spent in the routine inclusive of calls.
- REAL(SP) EXCLUSIVE_CPU_TIME
User CPU time spent in the routine exclusive of calls.
- REAL(SP) EXCLUSIVE_SYSTEM_TIME
System CPU time spent in the routine exclusive of calls.
- LOGICAL DIAGNOSTICS
.TRUE. if diagnostics are active in the routine
- LOGICAL TIMING
.TRUE. if timing is active in the routine
- TYPE(ROUTINE_LIST_ITEM_TYPE), pointer ROUTINE_LIST_ITEM
Pointer to the routine list item for diagnostics or timing.
- TYPE(ROUTINE_STACK_ITEM_TYPE), pointer PREVIOUS_ROUTINE
Pointer to the previous routine in the routine stack.

7.6.1 Detailed Description

Contains information for an item in the routine invocation stack.

Definition at line 126 of file base_routines.f90.

7.6.2 Member Data Documentation

7.6.2.1 LOGICAL BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::DIAGNOSTICS [private]

.TRUE. if diagnostics are active in the routine

Definition at line 132 of file base_routines.f90.

**7.6.2.2 REAL(SP) BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::EXCLUSIVE_CPU_-
TIME [private]**

User CPU time spent in the routine exclusive of calls.

Definition at line 130 of file base_routines.f90.

**7.6.2.3 REAL(SP) BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::EXCLUSIVE_-
SYSTEM_TIME [private]**

System CPU time spent in the routine exclusive of calls.

Definition at line 131 of file base_routines.f90.

**7.6.2.4 REAL(SP) BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::INCLUSIVE_CPU_-
TIME [private]**

User CPU time spent in the routine inclusive of calls.

Definition at line 128 of file base_routines.f90.

**7.6.2.5 REAL(SP) BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::INCLUSIVE_-
SYSTEM_TIME [private]**

System CPU time spent in the routine inclusive of calls.

Definition at line 129 of file base_routines.f90.

**7.6.2.6 TYPE(VARYING_STRING) BASE_ROUTINES::ROUTINE_STACK_ITEM_-
TYPE::NAME [private]**

The name of the routine.

Definition at line 127 of file base_routines.f90.

**7.6.2.7 TYPE(ROUTINE_STACK_ITEM_TYPE),pointer BASE_ROUTINES::ROUTINE_-
STACK_ITEM_TYPE::PREVIOUS_ROUTINE [private]**

Pointer to the previous routine in the routine stack.

Definition at line 135 of file base_routines.f90.

**7.6.2.8 TYPE(ROUTINE_LIST_ITEM_TYPE),pointer BASE_ROUTINES::ROUTINE_-
STACK_ITEM_TYPE::ROUTINE_LIST_ITEM [private]**

Pointer to the routine list item for diagnostics or timing.

Definition at line 134 of file base_routines.f90.

**7.6.2.9 LOGICAL BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE::TIMING
[private]**

.TRUE. if timing is active in the routine

Definition at line 133 of file base_routines.f90.

7.7 BASE_ROUTINES::ROUTINE_STACK_TYPE Struct Reference

Contains information for the routine invocation stack.

Collaboration diagram for BASE_ROUTINES::ROUTINE_STACK_TYPE:

Private Attributes

- TYPE(ROUTINE_STACK_ITEM_TYPE), pointer STACK_POINTER
Pointer to the top of the stack.

7.7.1 Detailed Description

Contains information for the routine invocation stack.

Definition at line 139 of file base_routines.f90.

7.7.2 Member Data Documentation

7.7.2.1 TYPE(ROUTINE_STACK_ITEM_TYPE),pointer BASE_ROUTINES::ROUTINE_STACK_TYPE::STACK_POINTER [private]

Pointer to the top of the stack.

Definition at line 140 of file base_routines.f90.

7.8 BINARY_FILE::BINARY_FILE_INFO_TYPE Struct Reference

Public Attributes

- INTEGER(INTG) FILE_NUMBER
- INTEGER(INTG) BINARY_FILE_REVISION
- INTEGER(INTG) MACHINE_TYPE
- INTEGER(INTG) OS_TYPE
- INTEGER(INTG) ENDIAN_TYPE
- INTEGER(INTG) CHAR_FORMAT
- INTEGER(INTG) INT_FORMAT
- INTEGER(INTG) SP_FORMAT
- INTEGER(INTG) DP_FORMAT
- INTEGER(INTG) CHARACTER_SIZE
- INTEGER(INTG) INTEGER_SIZE
- INTEGER(INTG) SINTEGER_SIZE
- INTEGER(INTG) LINTEGER_SIZE
- INTEGER(INTG) SP_REAL_SIZE
- INTEGER(INTG) DP_REAL_SIZE
- INTEGER(INTG) LOGICAL_SIZE
- INTEGER(INTG) SPC_REAL_SIZE
- INTEGER(INTG) DPC_REAL_SIZE
- CHARACTER(LEN=MAXSTRLEN) FILE_NAME
- INTEGER(INTG) ACCESS_TYPE

7.8.1 Detailed Description

Definition at line 263 of file binary_file_f.f90.

7.8.2 Member Data Documentation

7.8.2.1 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::ACCESS_TYPE

Definition at line 284 of file binary_file_f.f90.

7.8.2.2 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::BINARY_FILE_REVISION

Definition at line 266 of file binary_file_f.f90.

7.8.2.3 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::CHAR_FORMAT

Definition at line 270 of file binary_file_f.f90.

7.8.2.4 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::CHARACTER_SIZE

Definition at line 274 of file binary_file_f.f90.

7.8.2.5 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::DP_FORMAT

Definition at line 273 of file binary_file_f.f90.

7.8.2.6 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::DP_REAL_SIZE

Definition at line 279 of file binary_file_f.f90.

7.8.2.7 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::DPC_REAL_SIZE

Definition at line 282 of file binary_file_f.f90.

7.8.2.8 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::ENDIAN_TYPE

Definition at line 269 of file binary_file_f.f90.

7.8.2.9 CHARACTER(LEN=MAXSTRLEN) BINARY_FILE::BINARY_FILE_INFO_TYPE::FILE_NAME

Definition at line 283 of file binary_file_f.f90.

7.8.2.10 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::FILE_NUMBER

Definition at line 265 of file binary_file_f.f90.

7.8.2.11 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::INT_FORMAT

Definition at line 271 of file binary_file_f.f90.

7.8.2.12 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::INTEGER_SIZE

Definition at line 275 of file binary_file_f.f90.

7.8.2.13 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::LINTEGER_SIZE

Definition at line 277 of file binary_file_f.f90.

7.8.2.14 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::LOGICAL_SIZE

Definition at line 280 of file binary_file_f.f90.

7.8.2.15 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::MACHINE_TYPE

Definition at line 267 of file binary_file_f.f90.

7.8.2.16 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::OS_TYPE

Definition at line 268 of file binary_file_f.f90.

7.8.2.17 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::SINTEGER_SIZE

Definition at line 276 of file binary_file_f.f90.

7.8.2.18 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::SP_FORMAT

Definition at line 272 of file binary_file_f.f90.

7.8.2.19 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::SP_REAL_SIZE

Definition at line 278 of file binary_file_f.f90.

7.8.2.20 INTEGER(INTG) BINARY_FILE::BINARY_FILE_INFO_TYPE::SPC_REAL_SIZE

Definition at line 281 of file binary_file_f.f90.

7.9 BINARY_FILE::BINARY_FILE_TYPE Struct Reference

Collaboration diagram for BINARY_FILE::BINARY_FILE_TYPE:

Public Attributes

- TYPE(BINARY_FILE_INFO_TYPE), pointer FILE_INFORMATION

7.9.1 Detailed Description

Definition at line 287 of file binary_file_f.f90.

7.9.2 Member Data Documentation

7.9.2.1 TYPE(BINARY_FILE_INFO_TYPE),pointer BINARY_FILE::BINARY_FILE_TYPE::FILE_INFORMATION

Definition at line 288 of file binary_file_f.f90.

7.10 BINARY_FILE::BINARY_TAG_TYPE Struct Reference

Public Attributes

- INTEGER(INTG) [INDEX](#)
- INTEGER(INTG) [NUM_SUBTAGS](#)
- INTEGER(INTG) [NUM_BYTES](#)
- INTEGER(INTG) [NUM_HEADER_BYTES](#)
- CHARACTER(LEN=MAXSTRLEN) [HEADER](#)

7.10.1 Detailed Description

Definition at line 291 of file binary_file.f90.

7.10.2 Member Data Documentation

7.10.2.1 CHARACTER(LEN=MAXSTRLEN) BINARY_FILE::BINARY_TAG_TYPE::HEADER

Definition at line 296 of file binary_file.f90.

7.10.2.2 INTEGER(INTG) BINARY_FILE::BINARY_TAG_TYPE::INDEX

Definition at line 292 of file binary_file.f90.

7.10.2.3 INTEGER(INTG) BINARY_FILE::BINARY_TAG_TYPE::NUM_BYTES

Definition at line 294 of file binary_file.f90.

7.10.2.4 INTEGER(INTG) BINARY_FILE::BINARY_TAG_TYPE::NUM_HEADER_BYTES

Definition at line 295 of file binary_file.f90.

7.10.2.5 INTEGER(INTG) BINARY_FILE::BINARY_TAG_TYPE::NUM_SUBTAGS

Definition at line 293 of file binary_file.f90.

7.11 BINARY_FILE::interface Interface Reference

Public Member Functions

- subroutine **BINARYCLOSEFILE** (FILE_NUMBER, ERR, CERROR)
- subroutine **BINARYOPENFILE** (FILE_NUMBER, CFNAME, CACCESSCODE, ERR, CERROR)
- subroutine **BINARYSETFILE** (FILE_NUMBER, SET_CODE, ERR, CERROR)
- subroutine **BINARYSKIPFILE** (FILE_NUMBER, NUMBER_BYTES, ERR, CERROR)
- subroutine **ISBINARYFILEOPEN** (FILE_NUMBER, RETURNCODE, ERR, CERROR)
- subroutine **ISENDBINARYFILE** (FILE_NUMBER, RETURN_CODE, ERR, CERROR)

7.11.1 Detailed Description

Definition at line 305 of file binary_file.f90.

7.11.2 Member Function Documentation

**7.11.2.1 subroutine BINARY_FILE::interface::BINARYCLOSEFILE
(INTEGER(INTG),intent(in) FILE_NUMBER, INTEGER(INTG),intent(out) ERR,
INTEGER(INTG),dimension(*),intent(out) CERROR)**

Definition at line 307 of file binary_file.f90.

**7.11.2.2 subroutine BINARY_FILE::interface::BINARYOPENFILE (INTEGER(INTG),intent(in)
FILE_NUMBER, INTEGER(INTG),dimension(*),intent(in)
CFNAME, INTEGER(INTG),dimension(*),intent(in) CACCESSCODE,
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(*),intent(out)
CERROR)**

Definition at line 314 of file binary_file.f90.

**7.11.2.3 subroutine BINARY_FILE::interface::BINARYSETFILE (INTEGER(INTG),intent(in)
FILE_NUMBER, INTEGER(INTG),intent(in) SET_CODE,
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(*),intent(out)
CERROR)**

Definition at line 323 of file binary_file.f90.

**7.11.2.4 subroutine BINARY_FILE::interface::BINARYSKIPFILE (INTEGER(INTG),intent(in)
FILE_NUMBER, INTEGER(INTG),intent(in) NUMBER_BYTES,
INTEGER(INTG),intent(out) ERR, INTEGER(INTG),dimension(*),intent(out)
CERROR)**

Definition at line 330 of file binary_file.f90.

7.11.2.5 subroutine `BINARY_FILE::interface::ISBINARYFILEOPEN` (INTEGER(INTG),intent(in) *FILE_NUMBER*, INTEGER(INTG),intent(out) *RETURNCODE*, INTEGER(INTG),intent(out) *ERR*, INTEGER(INTG),dimension(*),intent(out) *CERROR*)

Definition at line 341 of file binary_file_f.f90.

7.11.2.6 subroutine `BINARY_FILE::interface::ISENDMEMORYFILE` (INTEGER(INTG),intent(in) *FILE_NUMBER*, INTEGER(INTG),intent(out) *RETURN_CODE*, INTEGER(INTG),intent(out) *ERR*, INTEGER(INTG),dimension(*),intent(out) *CERROR*)

Definition at line 348 of file binary_file_f.f90.

7.12 BINARY_FILE::READ_BINARY_FILE Interface Reference

Public Member Functions

- subroutine [READ_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_CHARACTER](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_LOGICAL1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_SPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [READ_BINARY_FILE_DPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

7.12.1 Detailed Description

Definition at line 357 of file binary_file_f.f90.

7.12.2 Member Function Documentation

**7.12.2.1 subroutine BINARY_FILE::READ_BINARY_FILE::READ_BINARY_-
FILE_CHARACTER (TYPE(BINARY_FILE_TYPE),intent(in) FILEID,
INTEGER(INTG),intent(in) NUM_DATA, CHARACTER(LEN=*),intent(out) DATA,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 1341 of file binary_file_f.f90.

**7.12.2.2 subroutine BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DP
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)
NUM_DATA, REAL(DP),dimension(*),intent(out) DATA, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 1243 of file binary_file_f.f90.

**7.12.2.3 subroutine BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DP1
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)
NUM_DATA, REAL(DP),intent(out) DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 1290 of file binary_file_f.f90.

7.12.2.4 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DPC`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, COMPLEX(DPC),dimension(*),intent(out) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1591 of file binary_file_f.f90.

7.12.2.5 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_DPC1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, COMPLEX(DPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1639 of file binary_file_f.f90.

7.12.2.6 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_INTG`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, INTEGER(INTG),dimension(*),intent(out) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 851 of file binary_file_f.f90.

7.12.2.7 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_INTG1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, INTEGER(INTG),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 898 of file binary_file_f.f90.

7.12.2.8 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_LINTG`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, INTEGER(LINTG),dimension(*),intent(out) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1047 of file binary_file_f.f90.

7.12.2.9 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_LINTG1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, INTEGER(LINTG),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1094 of file binary_file_f.f90.

7.12.2.10 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_-`
`FILE_LOGICAL (TYPE(BINARY_FILE_TYPE),intent(in) FILEID,`
`INTEGER(INTG),intent(in) NUM_DATA, LOGICAL,dimension(*),intent(out) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Definition at line 1393 of file binary_file_f.f90.

7.12.2.11 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_-
FILE_LOGICAL1** (TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*,
INTEGER(INTG),intent(in) *NUM_DATA*, LOGICAL,intent(out) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 1440 of file binary_file_f.f90.

7.12.2.12 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SINTG**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, INTEGER(SINTG),dimension(*),intent(out) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 949 of file binary_file_f.f90.

7.12.2.13 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SINTG1**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, INTEGER(SINTG),intent(out) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 996 of file binary_file_f.f90.

7.12.2.14 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SP**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, REAL(SP),dimension(*),intent(out) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 1145 of file binary_file_f.f90.

7.12.2.15 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SP1**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, REAL(SP),intent(out) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 1192 of file binary_file_f.f90.

7.12.2.16 subroutine **BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SPC**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, COMPLEX(SPC),dimension(*),intent(out) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 1491 of file binary_file_f.f90.

7.12.2.17 subroutine `BINARY_FILE::READ_BINARY_FILE::READ_BINARY_FILE_SPC1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)`
`NUM_DATA, COMPLEX(SPC),intent(out) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 1539 of file binary_file_f.f90.

7.13 BINARY_FILE::WRITE_BINARY_FILE Interface Reference

Public Member Functions

- subroutine `WRITE_BINARY_FILE_INTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_INTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_SPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `WRITE_BINARY_FILE_DPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

7.13.1 Detailed Description

Definition at line 377 of file binary_file_f.f90.

7.13.2 Member Function Documentation

**7.13.2.1 subroutine BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-
FILE_CHARACTER (TYPE(BINARY_FILE_TYPE),intent(in) FILEID,
INTEGER(INTG),intent(in) NUM_DATA, CHARACTER(LEN=*),intent(in) DATA,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2466 of file binary_file_f.f90.

**7.13.2.2 subroutine BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DP
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)
NUM_DATA, REAL(DP),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2376 of file binary_file_f.f90.

**7.13.2.3 subroutine BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DP1
(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in)
NUM_DATA, REAL(DP),intent(in) DATA, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Definition at line 2420 of file binary_file_f.f90.

7.13.2.4 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DPC`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, COMPLEX(DPC),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2689 of file binary_file_f.f90.

7.13.2.5 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_DPC1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, COMPLEX(DPC),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2733 of file binary_file_f.f90.

7.13.2.6 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_INTG`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, INTEGER(INTG),dimension(*),intent(in) DATA, INTEGER(INTG),intent(out)`
`ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2022 of file binary_file_f.f90.

7.13.2.7 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_INTG1`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, INTEGER(INTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2063 of file binary_file_f.f90.

7.13.2.8 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_LINTG`
`(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, INTEGER(LINTG),dimension(*),intent(in) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2197 of file binary_file_f.f90.

7.13.2.9 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_-`
LINTG1 `(TYPE(BINARY_FILE_TYPE),intent(in) FILEID, INTEGER(INTG),intent(in) NUM_-`
`DATA, INTEGER(LINTG),intent(in) DATA, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, *)`

Definition at line 2240 of file binary_file_f.f90.

7.13.2.10 subroutine `BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-`
FILE_LOGICAL `(TYPE(BINARY_FILE_TYPE),intent(in) FILEID,`
`INTEGER(INTG),intent(in) NUM_-DATA, LOGICAL,dimension(*),intent(in) DATA,`
`INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,`
`*)`

Definition at line 2510 of file binary_file_f.f90.

7.13.2.11 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-
FILE_LOGICAL1** (TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*,
INTEGER(INTG),intent(in) *NUM_DATA*, LOGICAL,intent(in) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 2552 of file binary_file_f90.

7.13.2.12 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-
FILE_SINTG** (TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*,
INTEGER(INTG),intent(in) *NUM_DATA*, INTEGER(SINTG),dimension(*),intent(in)
DATA, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)

Definition at line 2108 of file binary_file_f90.

7.13.2.13 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_-
FILE_SINTG1** (TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*,
INTEGER(INTG),intent(in) *NUM_DATA*, INTEGER(SINTG),intent(in) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 2151 of file binary_file_f90.

7.13.2.14 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_SP**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, REAL(SP),dimension(*),intent(in) *DATA*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 2286 of file binary_file_f90.

7.13.2.15 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_SP1**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, REAL(SP),intent(in) *DATA*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 2330 of file binary_file_f90.

7.13.2.16 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_SPC**
(TYPE(BINARY_FILE_TYPE),intent(in) *FILEID*, INTEGER(INTG),intent(in)
NUM_DATA, COMPLEX(SPC),dimension(*),intent(in) *DATA*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*)

Definition at line 2598 of file binary_file_f90.

7.13.2.17 subroutine **BINARY_FILE::WRITE_BINARY_FILE::WRITE_BINARY_FILE_SPC1**
(**TYPE(BINARY_FILE_TYPE),intent(in)** *FILEID*, **INTEGER(INTG),intent(in)**
NUM_DATA, **COMPLEX(SPC),intent(in)** *DATA*, **INTEGER(INTG),intent(out)** *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Definition at line 2642 of file binary_file_f.f90.

7.14 BLAS::interface Interface Reference

Public Member Functions

- subroutine [SASUM](#) (N, X, INCX)
- subroutine [DASUM](#) (N, X, INCX)
- subroutine [SAXPY](#) (N, A, X, INCX, Y, INCY)
- subroutine [DAXPY](#) (N, A, X, INCX, Y, INCY)
- subroutine [SCOPY](#) (N, DX, INCX, DY, INCY)
- subroutine [DCOPY](#) (N, DX, INCX, DY, INCY)
- REAL(SP) [SDOT](#) (N, X, INCX, Y, INCY)
- REAL(DP) [DDOT](#) (N, X, INCX, Y, INCY)
- REAL(SP) [SNRM2](#) (N, X, INCX)
- REAL(DP) [DNRM2](#) (N, X, INCX)
- subroutine [SROT](#) (N, DX, INCX, DY, INCY, C, S)
- subroutine [DROT](#) (N, DX, INCX, DY, INCY, C, S)
- subroutine [SROTG](#) (DA, DB, C, S)
- subroutine [DROTG](#) (DA, DB, C, S)
- subroutine [SSCAL](#) (N, A, X, INCX)
- subroutine [DSCAL](#) (N, A, X, INCX)
- subroutine [STRSV](#) (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)
- subroutine [DTRSV](#) (UPLO, TRANS, DIAG, N, A, LDA, X, INCX)

7.14.1 Detailed Description

Definition at line 50 of file blas.f90.

7.14.2 Member Function Documentation

7.14.2.1 subroutine BLAS::interface::DASUM (INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)

Definition at line 61 of file blas.f90.

7.14.2.2 subroutine BLAS::interface::DAXPY (INTEGER(INTG),intent(in) *N*, REAL(DP),intent(in) *A*, REAL(DP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*, REAL(DP),dimension(*),intent(out) *Y*, INTEGER(INTG),intent(in) *INCY*)

Definition at line 78 of file blas.f90.

7.14.2.3 subroutine BLAS::interface::DCOPY (INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(*),intent(in) *DX*, INTEGER(INTG),intent(in) *INCX*, REAL(DP),dimension(*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*)

Definition at line 97 of file blas.f90.

**7.14.2.4 REAL(DP) BLAS::interface::DDOT (INTEGER(INTG),intent(in) *N*,
REAL(DP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*,
REAL(DP),dimension(*),intent(in) *Y*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 116 of file blas.f90.

**7.14.2.5 REAL(DP) BLAS::interface::DNRM2 (INTEGER(INTG),intent(in) *N*,
REAL(DP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 134 of file blas.f90.

**7.14.2.6 subroutine BLAS::interface::DROT (INTEGER(INTG),intent(in) *N*,
REAL(DP),dimension(*),intent(out) *DX*, INTEGER(INTG),intent(in) *INCX*,
REAL(DP),dimension(*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*,
REAL(DP),intent(in) *C*, REAL(DP),intent(in) *S*)**

Definition at line 153 of file blas.f90.

**7.14.2.7 subroutine BLAS::interface::DROTG (REAL(DP),intent(in) *DA*, REAL(DP),intent(in) *DB*,
REAL(DP),intent(in) *C*, REAL(DP),intent(in) *S*)**

Definition at line 172 of file blas.f90.

**7.14.2.8 subroutine BLAS::interface::DSCAL (INTEGER(INTG),intent(in)
N, REAL(DP),intent(in) *A*, REAL(DP),dimension(*),intent(inout) *X*,
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 188 of file blas.f90.

**7.14.2.9 subroutine BLAS::interface::DTRSV (CHARACTER(LEN=1),intent(in) *UPLO*,
CHARACTER(LEN=1),intent(in) *TRANS*, CHARACTER(LEN=1),intent(in)
DIAG, INTEGER(INTG),intent(in) *N*, REAL(DP),dimension(*lda*, *),intent(in)
A, INTEGER(INTG),intent(in) *LDA*, REAL(DP),dimension(*),intent(in) *X*,
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 237 of file blas.f90.

**7.14.2.10 subroutine BLAS::interface::SASUM (INTEGER(INTG),intent(in) *N*,
REAL(SP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 54 of file blas.f90.

**7.14.2.11 subroutine BLAS::interface::SAXPY (INTEGER(INTG),intent(in)
N, REAL(SP),intent(in) *A*, REAL(SP),dimension(*),intent(in) *X*,
INTEGER(INTG),intent(in) *INCX*, REAL(SP),dimension(*),intent(out) *Y*,
INTEGER(INTG),intent(in) *INCY*)**

Definition at line 68 of file blas.f90.

**7.14.2.12 subroutine BLAS::interface::SCOPY (INTEGER(INTG),intent(in) *N*,
REAL(SP),dimension(*),intent(in) *DX*, INTEGER(INTG),intent(in) *INCX*,
REAL(SP),dimension(*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 88 of file blas.f90.

**7.14.2.13 REAL(SP) BLAS::interface::SDOT (INTEGER(INTG),intent(in) *N*,
REAL(SP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*,
REAL(SP),dimension(*),intent(in) *Y*, INTEGER(INTG),intent(in) *INCY*)**

Definition at line 106 of file blas.f90.

**7.14.2.14 REAL(SP) BLAS::interface::SNRM2 (INTEGER(INTG),intent(in) *N*,
REAL(SP),dimension(*),intent(in) *X*, INTEGER(INTG),intent(in) *INCX*)**

Definition at line 126 of file blas.f90.

**7.14.2.15 subroutine BLAS::interface::SROT (INTEGER(INTG),intent(in) *N*,
REAL(SP),dimension(*),intent(out) *DX*, INTEGER(INTG),intent(in) *INCX*,
REAL(SP),dimension(*),intent(out) *DY*, INTEGER(INTG),intent(in) *INCY*,
REAL(SP),intent(in) *C*, REAL(SP),intent(in) *S*)**

Definition at line 142 of file blas.f90.

**7.14.2.16 subroutine BLAS::interface::SROTG (REAL(SP),intent(in) *DA*, REAL(SP),intent(in)
DB, REAL(SP),intent(in) *C*, REAL(SP),intent(in) *S*)**

Definition at line 164 of file blas.f90.

**7.14.2.17 subroutine BLAS::interface::SSCAL (INTEGER(INTG),intent(in)
N, REAL(SP),intent(in) *A*, REAL(SP),dimension(*),intent(inout) *X*,
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 180 of file blas.f90.

**7.14.2.18 subroutine BLAS::interface::STRSV (CHARACTER(LEN=1),intent(in) *UPLO*,
CHARACTER(LEN=1),intent(in) *TRANS*, CHARACTER(LEN=1),intent(in)
DIAG, INTEGER(INTG),intent(in) *N*, REAL(SP),dimension(*lda*, *),intent(in)
A, INTEGER(INTG),intent(in) *LDA*, REAL(SP),dimension(*),intent(in) *X*,
INTEGER(INTG),intent(in) *INCX*)**

Definition at line 228 of file blas.f90.

7.15 CMISS_PARMETIS::interface Interface Reference

Public Member Functions

- subroutine [ParMETIS_V3_PartMeshKway](#)

Private Member Functions

- subroutine [ParMETIS_V3_PartK](#) (*vtxdist*, *xadj*, *adjncy*, *vwgt*, *adjwgt*, *wgtflag*, *numflag*, *ncon*, *nparts*, *tpwgts*, *ubvec*,*&options*, *edgecut*, *part*, *comm*)

7.15.1 Detailed Description

Definition at line 61 of file cmiss_parmetis.f90.

7.15.2 Member Function Documentation

7.15.2.1 subroutine CMISS_PARMETIS::interface::ParMETIS_V3_PartK
(INTEGER(INTG),dimension()* *vtxdist**, INTEGER(INTG),dimension(*)* *xadj**,*
INTEGER(INTG),dimension()* *adjncy**, INTEGER(INTG),dimension(*)* *vwgt**,*
INTEGER(INTG),dimension()* *adjwgt**, INTEGER(INTG)* *wgtflag**, INTEGER(INTG)*
*numflag**, INTEGER(INTG)* *ncon**, INTEGER(INTG)* *nparts**, REAL(SP),dimension(*)*
*tpwgts**, REAL(SP),dimension(*)* *ubvec**, &,dimension(*)* *options**, INTEGER(INTG)*
*edgecut**, INTEGER(INTG),dimension(*)* *part**, INTEGER(INTG)* *comm* *[private]*

Definition at line 63 of file cmiss_parmetis.f90.

7.15.2.2 subroutine CMISS_PARMETIS::interface::ParMETIS_V3_PartMeshKway ()

Definition at line 86 of file cmiss_parmetis.f90.

7.16 CMISS_PETSC::interface Interface Reference

Public Member Functions

- subroutine [SNESSetType](#) (snes, method, ierr)

Private Member Functions

- subroutine [ISDestroy](#) (indexset, ierr)
- subroutine [ISLocalToGlobalMappingApply](#) (ctx, type, nin, idxin, nout, idxout, ierr)
- subroutine [ISLocalToGlobalMappingApplyIS](#) (ctx, isin, isout, ierr)
- subroutine [ISLocalToGlobalMappingCreate](#) (comm, N, globalnum, ctx, ierr)
- subroutine [ISLocalToGlobalMappingDestroy](#) (ctx, ierr)
- subroutine [KSPCreate](#) (comm, ksp, ierr)
- subroutine [KSPDestroy](#) (ksp, ierr)
- subroutine [KSPGetConvergedReason](#) (ksp, reason, ierr)
- subroutine [KSPGetIterationNumber](#) (ksp, its, ierr)
- subroutine [KSPGetPC](#) (ksp, pc, ierr)
- subroutine [KSPGetResidualNorm](#) (ksp, rnorm, ierr)
- subroutine [KSPSetFromOptions](#) (ksp, ierr)
- subroutine [KSPSetOperators](#) (ksp, Amat, Pmat, flag, ierr)
- subroutine [KSPSetTolerances](#) (ksp, rtol, atol, dtol, maxits, ierr)
- subroutine [KSPSetType](#) (ksp, method, ierr)
- subroutine [KSPSetUp](#) (ksp, ierr)
- subroutine [KSPSolve](#) (ksp, b, x, ierr)
- subroutine [MatAssemblyBegin](#) (A, assemblytype, ierr)
- subroutine [MatAssemblyEnd](#) (A, assemblytype, ierr)
- subroutine [MatCreate](#) (comm, A, ierr)
- subroutine [MatCreateMPIAIJ](#) (co m, localm, localn, globalm, globaln, diagnumbervzperrow, diagnumbervzeachrow, offdiagnumbervzperrow, &offdiagnumbervzeachrow, A, ierr)
- subroutine [MatCreateMPIDense](#) (comm, localm, localn, globalm, globaln, matrixdata, A, ierr)
- subroutine [MatCreateSeqAIJ](#) (comm, m, n, numbernzperrow, numbernzeachrow, A, ierr)
- subroutine [MatCreateSeqDense](#) (comm, m, n, matrixdata, A, ierr)
- subroutine [MatDestroy](#) (A, ierr)
- subroutine [MatGetArray](#) (A, mat_data, mat_offset, ierr)
- subroutine [MatGetArrayF90](#) (A, mat_data, ierr)
- subroutine [MatGetOwnershipRange](#) (A, firstrow, lastrow, ierr)
- subroutine [MatGetValues](#) (A, m, idxm, n, idxn, values, ierr)
- subroutine [MatRestoreArray](#) (A, mat_data, mat_offset, ierr)
- subroutine [MatRestoreArrayF90](#) (A, mat_data, ierr)
- subroutine [MatSetLocalToGlobalMapping](#) (A, ctx, ierr)
- subroutine [MatSetOption](#) (A, option, ierr)
- subroutine [MatSetSizes](#) (A, localm, localn, globalM, globalN, ierr)
- subroutine [MatSetValue](#) (A, row, col, value, insertmode, ierr)
- subroutine [MatSetValues](#) (A, m, mindices, n, nindices, values, insertmode, ierr)
- subroutine [MatSetValuesLocal](#) (A, m, mindices, n, nindices, values, insertmode, ierr)
- subroutine [MatSetValueLocal](#) (A, row, col, value, insertmode, ierr)
- subroutine [MatView](#) (A, v, ierr)
- subroutine [MatZeroEntries](#) (A, ierr)
- subroutine [PCSetType](#) (pc, method, ierr)

- subroutine `PetscFinalize` (ierr)
- subroutine `PetscInitialize` (file, ierr)
- subroutine `PetscLogPrintSummary` (comm, file, ierr)
- subroutine `SNESCreate` (comm, snes, ierr)
- subroutine `SNESDestroy` (snes, ierr)
- subroutine `SNESSetFromOptions` (snes, ierr)
- subroutine `SNESSetFunction` (snes, f, ffunction, ctx, ierr)
- subroutine `SNESolve` (snes, b, x, ierr)
- subroutine `VecAssemblyBegin` (x, ierr)
- subroutine `VecAssemblyEnd` (x, ierr)
- subroutine `VecCreate` (comm, x, ierr)
- subroutine `VecCreateGhost` (comm, localm, globalm, nghost, ghosts, x, ierr)
- subroutine `VecCreateGhostWithArray` (comm, localm, globalm, nghost, ghosts, array, x, ierr)
- subroutine `VecCreateMPI` (comm, localm, globalm, x, ierr)
- subroutine `VecCreateMPIWithArray` (comm, localn, globaln, array, x, ierr)
- subroutine `VecCreateSeq` (comm, m, x, ierr)
- subroutine `VecCreateSeqWithArray` (comm, n, array, x, ierr)
- subroutine `VecDestroy` (x, ierr)
- subroutine `VecDuplicate` (old, new, ierr)
- subroutine `VecGetArray` (x, vec_data, vec_offset, ierr)
- subroutine `VecGetArrayF90` (x, vec_data, ierr)
- subroutine `VecGetLocalSize` (x, size, ierr)
- subroutine `VecGetOwnershipRange` (x, low, high, ierr)
- subroutine `VecGetSize` (x, size, ierr)
- subroutine `VecGetValues` (x, n, indices, values, ierr)
- subroutine `VecGhostGetLocalForm` (g, l, ierr)
- subroutine `VecGhostRestoreLocalForm` (g, l, ierr)
- subroutine `VecGhostUpdateBegin` (x, insertmode, scattermode, ierr)
- subroutine `VecGhostUpdateEnd` (x, insertmode, scattermode, ierr)
- subroutine `VecRestoreArray` (x, vec_data, vec_offset, ierr)
- subroutine `VecRestoreArrayF90` (x, vec_data, ierr)
- subroutine `VecSet` (x, value, ierr)
- subroutine `VecSetFromOptions` (x, ierr)
- subroutine `VecSetLocalToGlobalMapping` (v, ctx, ierr)
- subroutine `VecSetSizes` (x, localm, globalm, ierr)
- subroutine `VecSetValues` (x, n, indices, values, insertmode, ierr)
- subroutine `VecSetValuesLocal` (x, n, indices, values, insertmode, ierr)
- subroutine `VecView` (x, v, ierr)

7.16.1 Detailed Description

Definition at line 168 of file cmiss_petsc.f90.

7.16.2 Member Function Documentation

7.16.2.1 subroutine `CMISS_PETSC::interface::ISDestroy` (`indexset, ierr`) [private]

Definition at line 170 of file cmiss_petsc.f90.

7.16.2.2 subroutine CMISS_PETSC::interface::ISLocalToGlobalMappingApply (ctx, type, nin, idxin, nout, idxout, ierr) [private]

Definition at line 175 of file cmiss_petsc.f90.

7.16.2.3 subroutine CMISS_PETSC::interface::ISLocalToGlobalMappingApplyIS (ctx, isin, isout, ierr) [private]

Definition at line 185 of file cmiss_petsc.f90.

7.16.2.4 subroutine CMISS_PETSC::interface::ISLocalToGlobalMappingCreate (comm, N, globalnum, ctx, ierr) [private]

Definition at line 192 of file cmiss_petsc.f90.

7.16.2.5 subroutine CMISS_PETSC::interface::ISLocalToGlobalMappingDestroy (ctx, ierr) [private]

Definition at line 200 of file cmiss_petsc.f90.

7.16.2.6 subroutine CMISS_PETSC::interface::KSPCreate (comm, ksp, ierr) [private]

Definition at line 205 of file cmiss_petsc.f90.

7.16.2.7 subroutine CMISS_PETSC::interface::KSPDestroy (ksp, ierr) [private]

Definition at line 211 of file cmiss_petsc.f90.

7.16.2.8 subroutine CMISS_PETSC::interface::KSPGetConvergedReason (ksp, reason, ierr) [private]

Definition at line 216 of file cmiss_petsc.f90.

7.16.2.9 subroutine CMISS_PETSC::interface::KSPGetIterationNumber (ksp, its, ierr) [private]

Definition at line 222 of file cmiss_petsc.f90.

7.16.2.10 subroutine CMISS_PETSC::interface::KSPGetPC (ksp, pc, ierr) [private]

Definition at line 228 of file cmiss_petsc.f90.

7.16.2.11 subroutine CMISS_PETSC::interface::KSPGetResidualNorm (ksp, rnorm, ierr) [private]

Definition at line 234 of file cmiss_petsc.f90.

7.16.2.12 subroutine CMISS_PETSC::interface::KSPSetFromOptions (ksp, ierr) [private]

Definition at line 240 of file cmiss_petsc.f90.

7.16.2.13 subroutine CMISS_PETSC::interface::KSPSetOperators (ksp, Amat, Pmat, flag, ierr) [private]

Definition at line 245 of file cmiss_petsc.f90.

7.16.2.14 subroutine CMISS_PETSC::interface::KSPSetTolerances (ksp, rtol, atol, dtol, maxits, ierr) [private]

Definition at line 253 of file cmiss_petsc.f90.

7.16.2.15 subroutine CMISS_PETSC::interface::KSPSetType (ksp, method, ierr) [private]

Definition at line 262 of file cmiss_petsc.f90.

7.16.2.16 subroutine CMISS_PETSC::interface::KSPSetUp (ksp, ierr) [private]

Definition at line 268 of file cmiss_petsc.f90.

7.16.2.17 subroutine CMISS_PETSC::interface::KSPSolve (ksp, b, x, ierr) [private]

Definition at line 273 of file cmiss_petsc.f90.

7.16.2.18 subroutine CMISS_PETSC::interface::MatAssemblyBegin (A, assemblytype, ierr) [private]

Definition at line 280 of file cmiss_petsc.f90.

7.16.2.19 subroutine CMISS_PETSC::interface::MatAssemblyEnd (A, assemblytype, ierr) [private]

Definition at line 286 of file cmiss_petsc.f90.

7.16.2.20 subroutine CMISS_PETSC::interface::MatCreate (comm, A, ierr) [private]

Definition at line 292 of file cmiss_petsc.f90.

7.16.2.21 subroutine CMISS_PETSC::interface::MatCreateMPIAIJ (co m, localm, localn, globalm, globaln, diagnumbervzperrow, diagnumbervzeachrow, offdiagnumbervzperrow, & offdiagnumbervzeachrow, A, ierr) [private]

Definition at line 298 of file cmiss_petsc.f90.

7.16.2.22 subroutine CMISS_PETSC::interface::MatCreateMPIDense (comm, localm, localn, globalm, globaln, matrixdata, A, ierr) [private]

Definition at line 313 of file cmiss_petsc.f90.

7.16.2.23 subroutine CMISS_PETSC::interface::MatCreateSeqAIJ (comm, m, n, numbernzperrow, numbernzeachrow, A, ierr) [private]

Definition at line 324 of file cmiss_petsc.f90.

7.16.2.24 subroutine CMISS_PETSC::interface::MatCreateSeqDense (comm, m, n, matrixdata, A, ierr) [private]

Definition at line 334 of file cmiss_petsc.f90.

7.16.2.25 subroutine CMISS_PETSC::interface::MatDestroy (A, ierr) [private]

Definition at line 343 of file cmiss_petsc.f90.

7.16.2.26 subroutine CMISS_PETSC::interface::MatGetArray (A, mat_data, mat_offset, ierr) [private]

Definition at line 348 of file cmiss_petsc.f90.

7.16.2.27 subroutine CMISS_PETSC::interface::MatGetArrayF90 (A, mat_data, ierr) [private]

Definition at line 355 of file cmiss_petsc.f90.

7.16.2.28 subroutine CMISS_PETSC::interface::MatGetOwnershipRange (A, firstrow, lastrow, ierr) [private]

Definition at line 361 of file cmiss_petsc.f90.

7.16.2.29 subroutine CMISS_PETSC::interface::MatGetValues (A, m, idxm, n, idxn, values, ierr) [private]

Definition at line 368 of file cmiss_petsc.f90.

7.16.2.30 subroutine CMISS_PETSC::interface::MatRestoreArray (A, mat_data, mat_offset, ierr) [private]

Definition at line 378 of file cmiss_petsc.f90.

7.16.2.31 subroutine CMISS_PETSC::interface::MatRestoreArrayF90 (A, mat_data, ierr) [private]

Definition at line 385 of file cmiss_petsc.f90.

7.16.2.32 subroutine CMISS_PETSC::interface::MatSetLocalToGlobalMapping (A, ctx, ierr) [private]

Definition at line 391 of file cmiss_petsc.f90.

7.16.2.33 subroutine CMISS_PETSC::interface::MatSetOption (A, option, ierr) [private]

Definition at line 397 of file cmiss_petsc.f90.

7.16.2.34 subroutine CMISS_PETSC::interface::MatSetSizes (A, localm, localn, globalM, globalN, ierr) [private]

Definition at line 403 of file cmiss_petsc.f90.

7.16.2.35 subroutine CMISS_PETSC::interface::MatSetValue (A, row, col, value, insertmode, ierr) [private]

Definition at line 412 of file cmiss_petsc.f90.

7.16.2.36 subroutine CMISS_PETSC::interface::MatSetValueLocal (A, row, col, value, insertmode, ierr) [private]

Definition at line 443 of file cmiss_petsc.f90.

7.16.2.37 subroutine CMISS_PETSC::interface::MatSetValues (A, m, mindices, n, nindices, values, insertmode, ierr) [private]

Definition at line 421 of file cmiss_petsc.f90.

7.16.2.38 subroutine CMISS_PETSC::interface::MatSetValuesLocal (A, m, mindices, n, nindices, values, insertmode, ierr) [private]

Definition at line 432 of file cmiss_petsc.f90.

7.16.2.39 subroutine CMISS_PETSC::interface::MatView (A, v, ierr) [private]

Definition at line 452 of file cmiss_petsc.f90.

7.16.2.40 subroutine CMISS_PETSC::interface::MatZeroEntries (A, ierr) [private]

Definition at line 458 of file cmiss_petsc.f90.

7.16.2.41 subroutine CMISS_PETSC::interface::PCSetType (pc, method, ierr) [private]

Definition at line 463 of file cmiss_petsc.f90.

7.16.2.42 subroutine CMISS_PETSC::interface::PetscFinalize (ierr) [private]

Definition at line 469 of file cmiss_petsc.f90.

7.16.2.43 subroutine CMISS_PETSC::interface::PetscInitialize (file, ierr) [private]

Definition at line 473 of file cmiss_petsc.f90.

7.16.2.44 subroutine CMISS_PETSC::interface::PetscLogPrintSummary (comm, file, ierr) [private]

Definition at line 478 of file cmiss_petsc.f90.

7.16.2.45 subroutine CMISS_PETSC::interface::SNESCreate (comm, snes, ierr) [private]

Definition at line 484 of file cmiss_petsc.f90.

7.16.2.46 subroutine CMISS_PETSC::interface::SNESDestroy (snes, ierr) [private]

Definition at line 490 of file cmiss_petsc.f90.

7.16.2.47 subroutine CMISS_PETSC::interface::SNESSetFromOptions (snes, ierr) [private]

Definition at line 495 of file cmiss_petsc.f90.

7.16.2.48 subroutine CMISS_PETSC::interface::SNESSetFunction (snes, f, ffunction, TYPE(SOLUTION_TYPE),pointer ctx, ierr) [private]

Definition at line 500 of file cmiss_petsc.f90.

7.16.2.49 subroutine CMISS_PETSC::interface::SNESGetType (snes, method, ierr)

Definition at line 509 of file cmiss_petsc.f90.

7.16.2.50 subroutine CMISS_PETSC::interface::SNESolve (snes, b, x, ierr) [private]

Definition at line 515 of file cmiss_petsc.f90.

7.16.2.51 subroutine CMISS_PETSC::interface::VecAssemblyBegin (x, ierr) [private]

Definition at line 522 of file cmiss_petsc.f90.

7.16.2.52 subroutine CMISS_PETSC::interface::VecAssemblyEnd (x, ierr) [private]

Definition at line 527 of file cmiss_petsc.f90.

7.16.2.53 subroutine CMISS_PETSC::interface::VecCreate (comm, x, ierr) [private]

Definition at line 532 of file cmiss_petsc.f90.

7.16.2.54 subroutine CMISS_PETSC::interface::VecCreateGhost (comm, localm, globalm, nghost, ghosts, x, ierr) [private]

Definition at line 538 of file cmiss_petsc.f90.

7.16.2.55 subroutine CMISS_PETSC::interface::VecCreateGhostWithArray (comm, localm, globalm, nghost, ghosts, array, x, ierr) [private]

Definition at line 548 of file cmiss_petsc.f90.

7.16.2.56 subroutine CMISS_PETSC::interface::VecCreateMPI (comm, localm, globalm, x, ierr) [private]

Definition at line 559 of file cmiss_petsc.f90.

7.16.2.57 subroutine CMISS_PETSC::interface::VecCreateMPIWithArray (comm, localn, globaln, array, x, ierr) [private]

Definition at line 567 of file cmiss_petsc.f90.

7.16.2.58 subroutine CMISS_PETSC::interface::VecCreateSeq (comm, m, x, ierr) [private]

Definition at line 576 of file cmiss_petsc.f90.

7.16.2.59 subroutine CMISS_PETSC::interface::VecCreateSeqWithArray (comm, n, array, x, ierr) [private]

Definition at line 583 of file cmiss_petsc.f90.

7.16.2.60 subroutine CMISS_PETSC::interface::VecDestroy (x, ierr) [private]

Definition at line 591 of file cmiss_petsc.f90.

7.16.2.61 subroutine CMISS_PETSC::interface::VecDuplicate (old, new, ierr) [private]

Definition at line 596 of file cmiss_petsc.f90.

7.16.2.62 subroutine CMISS_PETSC::interface::VecGetArray (x, vec_data, vec_offset, ierr) [private]

Definition at line 601 of file cmiss_petsc.f90.

7.16.2.63 subroutine CMISS_PETSC::interface::VecGetArrayF90 (x, vec_data, ierr)
[private]

Definition at line 608 of file cmiss_petsc.f90.

7.16.2.64 subroutine CMISS_PETSC::interface::VecGetLocalSize (x, size, ierr) [private]

Definition at line 614 of file cmiss_petsc.f90.

7.16.2.65 subroutine CMISS_PETSC::interface::VecGetOwnershipRange (x, low, high, ierr)
[private]

Definition at line 620 of file cmiss_petsc.f90.

7.16.2.66 subroutine CMISS_PETSC::interface::VecGetSize (x, size, ierr) [private]

Definition at line 627 of file cmiss_petsc.f90.

7.16.2.67 subroutine CMISS_PETSC::interface::VecGetValues (x, n, indices, values, ierr)
[private]

Definition at line 633 of file cmiss_petsc.f90.

7.16.2.68 subroutine CMISS_PETSC::interface::VecGhostGetLocalForm (g, l, ierr)
[private]

Definition at line 641 of file cmiss_petsc.f90.

7.16.2.69 subroutine CMISS_PETSC::interface::VecGhostRestoreLocalForm (g, l, ierr)
[private]

Definition at line 647 of file cmiss_petsc.f90.

7.16.2.70 subroutine CMISS_PETSC::interface::VecGhostUpdateBegin (x, insertmode, scattermode, ierr) [private]

Definition at line 653 of file cmiss_petsc.f90.

7.16.2.71 subroutine CMISS_PETSC::interface::VecGhostUpdateEnd (x, insertmode, scattermode, ierr) [private]

Definition at line 660 of file cmiss_petsc.f90.

7.16.2.72 subroutine CMISS_PETSC::interface::VecRestoreArray (x, vec_data, vec_offset, ierr)
[private]

Definition at line 667 of file cmiss_petsc.f90.

7.16.2.73 subroutine CMISS_PETSC::interface::VecRestoreArrayF90 (x, vec_data, ierr)
[private]

Definition at line 674 of file cmiss_petsc.f90.

7.16.2.74 subroutine CMISS_PETSC::interface::VecSet (x, value, ierr) [private]

Definition at line 680 of file cmiss_petsc.f90.

7.16.2.75 subroutine CMISS_PETSC::interface::VecSetFromOptions (x, ierr) [private]

Definition at line 686 of file cmiss_petsc.f90.

7.16.2.76 subroutine CMISS_PETSC::interface::VecSetLocalToGlobalMapping (v, ctx, ierr)
[private]

Definition at line 691 of file cmiss_petsc.f90.

7.16.2.77 subroutine CMISS_PETSC::interface::VecSetSizes (x, localm, globalm, ierr)
[private]

Definition at line 697 of file cmiss_petsc.f90.

7.16.2.78 subroutine CMISS_PETSC::interface::VecSetValues (x, n, indices, values, insertmode,
ierr) [private]

Definition at line 703 of file cmiss_petsc.f90.

7.16.2.79 subroutine CMISS_PETSC::interface::VecSetValuesLocal (x, n, indices, values,
insertmode, ierr) [private]

Definition at line 712 of file cmiss_petsc.f90.

7.16.2.80 subroutine CMISS_PETSC::interface::VecView (x, v, ierr) [private]

Definition at line 721 of file cmiss_petsc.f90.

7.17 CMISS_PETSC_TYPES::PETSC_IS_TYPE Struct Reference

7.17.1 Detailed Description

Definition at line 65 of file cmiss_petsc_types.f90.

7.18 CMISS_PETSC_TYPES::PETSC_ISLOCALTOGLOBALMAPPING_- TYPE Struct Reference

7.18.1 Detailed Description

Definition at line 69 of file cmiss_petsc_types.f90.

7.19 CMISS_PETSC_TYPES::PETSC_KSP_TYPE Struct Reference

7.19.1 Detailed Description

Definition at line 73 of file cmiss_petsc_types.f90.

7.20 CMISS_PETSC_TYPES::PETSC_MAT_TYPE Struct Reference

7.20.1 Detailed Description

Definition at line 77 of file cmiss_petsc_types.f90.

7.21 CMISS_PETSC_TYPES::PETSC_PC_TYPE Struct Reference

7.21.1 Detailed Description

Definition at line 83 of file cmiss_petsc_types.f90.

7.22 CMISS_PETSC_TYPES::PETSC_SNES_TYPE Struct Reference

7.22.1 Detailed Description

Definition at line 87 of file cmiss_petsc_types.f90.

7.23 CMISS_PETSC_TYPES::PETSC_VEC_TYPE Struct Reference

7.23.1 Detailed Description

Definition at line 91 of file cmiss_petsc_types.f90.

7.24 COMP_ENVIRONMENT::CACHE_TYPE Struct Reference

Contains information on a cache hierarchy.

Private Attributes

- INTEGER(INTG) **NUMBER_LEVELS**
The number of levels in the cache hierarchy.
- INTEGER(INTG), allocatable **SIZE**
SIZE(level_idx). The size of the level_idx'th cache level.

7.24.1 Detailed Description

Contains information on a cache hierarchy.

Definition at line 64 of file computational_environment.f90.

7.24.2 Member Data Documentation

7.24.2.1 INTEGER(INTG) COMP_ENVIRONMENT::CACHE_TYPE::NUMBER_LEVELS [private]

The number of levels in the cache hierarchy.

Definition at line 65 of file computational_environment.f90.

7.24.2.2 INTEGER(INTG),allocatable COMP_ENVIRONMENT::CACHE_TYPE::SIZE [private]

SIZE(level_idx). The size of the level_idx'th cache level.

Definition at line 66 of file computational_environment.f90.

7.25 COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE Struct Reference

Contains information on the computational environment the program is running in.

Collaboration diagram for COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE:

Private Attributes

- INTEGER(INTG) [MPI_COMM](#)
The MPI communicator for cmiss.
- INTEGER(INTG) [NUMBER_COMPUTATIONAL_NODES](#)
The number of computational nodes.
- INTEGER(INTG) [MY_COMPUTATIONAL_NODE_NUMBER](#)
The index of the running process.
- TYPE([COMPUTATIONAL_NODE_TYPE](#)), allocatable [COMPUTATIONAL_NODES](#)
COMPUTATIONAL_NODES(node_idx). Contains information on the node_idx'th computational node.

7.25.1 Detailed Description

Contains information on the computational environment the program is running in.

Definition at line 88 of file computational_environment.f90.

7.25.2 Member Data Documentation

7.25.2.1 TYPE(COMPUTATIONAL_NODE_TYPE),allocatable COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE::COMPUTATIONAL_NODES [private]

COMPUTATIONAL_NODES(node_idx). Contains information on the node_idx'th computational node.

Definition at line 92 of file computational_environment.f90.

7.25.2.2 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE::MPI_COMM [private]

The MPI communicator for cmiss.

Definition at line 89 of file computational_environment.f90.

7.25.2.3 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE::MY_COMPUTATIONAL_NODE_NUMBER [private]

The index of the running process.

Definition at line 91 of file computational_environment.f90.

**7.25.2.4 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_-
ENVIRONMENT_TYPE::NUMBER_COMPUTATIONAL_NODES
[private]**

The number of computational nodes.

Definition at line 90 of file computational_environment.f90.

7.26 COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE Struct Reference

Contains information on a computational node containing a number of processors.

Private Attributes

- INTEGER(INTG) **NUMBER_PROCESSORS**
The number of processors for this computational node.
- INTEGER(INTG) **RANK**
The MPI rank of this computational node.
- INTEGER(INTG) **NODE_NAME_LENGTH**
The length of the name of the computational node.
- CHARACTER(LEN=MPI_MAX_PROCESSOR_NAME) **NODE_NAME**
The name of the computational node.

7.26.1 Detailed Description

Contains information on a computational node containing a number of processors.

Definition at line 70 of file computational_environment.f90.

7.26.2 Member Data Documentation

7.26.2.1 CHARACTER(LEN=MPI_MAX_PROCESSOR_NAME) COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE::NODE_NAME [private]

The name of the computational node.

Definition at line 75 of file computational_environment.f90.

7.26.2.2 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE::NODE_NAME_LENGTH [private]

The length of the name of the computational node.

Definition at line 74 of file computational_environment.f90.

7.26.2.3 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE::NUMBER_PROCESSORS [private]

The number of processors for this computational node.

Definition at line 71 of file computational_environment.f90.

**7.26.2.4 INTEGER(INTG) COMP_ENVIRONMENT::COMPUTATIONAL_NODE_-
TYPE::RANK [private]**

The MPI rank of this computational node.

Definition at line 72 of file computational_environment.f90.

7.27 COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE Struct Reference

Contains information on the MPI type to transfer information about a computational node.

Private Attributes

- INTEGER(INTG) [MPI_TYPE](#)
The MPI data type.
- INTEGER(INTG) [NUM_BLOCKS](#)
The number of blocks in the MPI data type. This will be equal to 4.
- INTEGER(INTG) [BLOCK_LENGTHS](#)
The length of each block.
- INTEGER(INTG) [TYPES](#)
The data types of each block.
- INTEGER(MPI_ADDRESS_KIND) [DISPLACEMENTS](#)
The address displacements to each block.

7.27.1 Detailed Description

Contains information on the MPI type to transfer information about a computational node.

Definition at line 79 of file computational_environment.f90.

7.27.2 Member Data Documentation

7.27.2.1 INTEGER(INTG) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE::BLOCK_LENGTHS [private]

The length of each block.

Definition at line 82 of file computational_environment.f90.

7.27.2.2 INTEGER(MPI_ADDRESS_KIND) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE::DISPLACEMENTS [private]

The address displacements to each block.

Definition at line 84 of file computational_environment.f90.

7.27.2.3 INTEGER(INTG) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE::MPI_TYPE [private]

The MPI data type.

Definition at line 80 of file computational_environment.f90.

**7.27.2.4 INTEGER(INTG) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_-
TYPE::NUM_BLOCKS [private]**

The number of blocks in the MPI data type. This will be equal to 4.

Definition at line 81 of file computational_environment.f90.

**7.27.2.5 INTEGER(INTG) COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_-
TYPE::TYPES [private]**

The data types of each block.

Definition at line 83 of file computational_environment.f90.

7.28 COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC Interface Reference

Private Member Functions

- REAL(DP) [COORDINATE_CONVERT_FROM_RC_DP](#) (COORDINATE_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE_CONVERT_FROM_RC_SP](#) (COORDINATE_SYSTEM, Z, ERR, ERROR)

7.28.1 Detailed Description

Definition at line 119 of file coordinate_routines.f90.

7.28.2 Member Function Documentation

7.28.2.1 REAL(DP) COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC::COORDINATE_CONVERT_FROM_RC_DP
(**TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,**
REAL(DP),dimension(:),intent(in) Z, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 227 of file coordinate_routines.f90.

7.28.2.2 REAL(SP) COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC::COORDINATE_CONVERT_FROM_RC_SP
(**TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM,**
REAL(SP),dimension(:),intent(in) Z, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 352 of file coordinate_routines.f90.

7.29 COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC Interface Reference

Private Member Functions

- REAL(DP) [COORDINATE_CONVERT_TO_RC_DP](#) (COORDINATE_SYSTEM, X, ERR, ERROR)
- REAL(SP) [COORDINATE_CONVERT_TO_RC_SP](#) (COORDINATE_SYSTEM, X, ERR, ERROR)

7.29.1 Detailed Description

Definition at line 124 of file coordinate_routines.f90.

7.29.2 Member Function Documentation

7.29.2.1 REAL(DP) COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC::COORDINATE_CONVERT_TO_RC_DP
(**TYPE(COORDINATE_SYSTEM_TYPE),intent(in)** *COORDINATE_SYSTEM*,
REAL(DP),dimension(:),intent(in) *X*, **INTEGER(INTG),intent(out)** *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 488 of file coordinate_routines.f90.

7.29.2.2 REAL(SP) COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC::COORDINATE_CONVERT_TO_RC_SP
(**TYPE(COORDINATE_SYSTEM_TYPE),intent(in)** *COORDINATE_SYSTEM*,
REAL(SP),dimension(:),intent(in) *X*, **INTEGER(INTG),intent(out)** *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 571 of file coordinate_routines.f90.

7.30 COORDINATE_ROUTINES::COORDINATE_DELTA_- CALCULATE Interface Reference

Private Member Functions

- REAL(DP) [COORDINATE_DELTA_CALCULATE_DP](#) (COORDINATE_SYSTEM, X, Y, ERR, ERROR)

7.30.1 Detailed Description

Definition at line 129 of file coordinate_routines.f90.

7.30.2 Member Function Documentation

- 7.30.2.1 REAL(DP) COORDINATE_ROUTINES::COORDINATE_DELTA_-
CALCULATE::COORDINATE_DELTA_CALCULATE_DP**
(TYPE(COORDINATE_SYSTEM_TYPE),intent(in) *COORDINATE_SYSTEM*,
REAL(DP),dimension(:,),intent(in) *X*, REAL(DP),dimension(:,),intent(in) *Y*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*)
[private]

Definition at line 664 of file coordinate_routines.f90.

7.31 COORDINATE_ROUTINES::COORDINATE_- DERIVATIVE_CONVERT_TO_RC Interface Reference

Private Member Functions

- COORDINATE_DERIVATIVE_CONVERT_TO_RC_DP
- COORDINATE_DERIVATIVE_CONVERT_TO_RC_SP

7.31.1 Detailed Description

Definition at line 182 of file coordinate_routines.f90.

7.31.2 Member Function Documentation

- 7.31.2.1 COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_-
TO_RC::COORDINATE_DERIVATIVE_CONVERT_TO_RC_DP 0
[private]
- 7.31.2.2 COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_-
TO_RC::COORDINATE_DERIVATIVE_CONVERT_TO_RC_SP 0
[private]

7.32 COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY Interface Reference

Private Member Functions

- subroutine **COORDINATE_SYSTEM_DESTROY_NUMBER** (USER_NUMBER, ERR,
ERROR,*)
- subroutine **COORDINATE_SYSTEM_DESTROY_PTR** (COORDINATE_SYSTEM, ERR,
ERROR,*)

7.32.1 Detailed Description

Definition at line 187 of file coordinate_routines.f90.

7.32.2 Member Function Documentation

7.32.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY::COORDINATE_SYSTEM_DESTROY_NUMBER
(**INTEGER(INTG) USER_NUMBER, INTEGER(INTG),intent(out) ERR,**
TYPE(VARYING_STRING),intent(out) ERROR, */ [private])

Definition at line 1818 of file coordinate_routines.f90.

7.32.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY::COORDINATE_SYSTEM_DESTROY_PTR
(**TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,**
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, */
[private])

Definition at line 1874 of file coordinate_routines.f90.

7.33 COORDINATE_ROUTINES::COORDINATE_SYSTEM_- DIMENSION_SET Interface Reference

Private Member Functions

- subroutine `COORDINATE_SYSTEM_DIMENSION_SET_NUMBER` (`USER_NUMBER`, `DIMENSION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_DIMENSION_SET_PTR` (`COORDINATE_SYSTEM`, `DIMENSION`, `ERR`, `ERROR,*`)

7.33.1 Detailed Description

Definition at line 134 of file coordinate_routines.f90.

7.33.2 Member Function Documentation

7.33.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_- DIMENSION_SET::COORDINATE_SYSTEM_DIMENSION_SET_NUMBER (`INTEGER(INTG),intent(in)` `USER_NUMBER`, `INTEGER(INTG),intent(in)` `DIMENSION`, `INTEGER(INTG),intent(out)` `ERR`, `TYPE(VARYING_-` `STRING),intent(out)` `ERROR`, `*`) [private]

Definition at line 1180 of file coordinate_routines.f90.

7.33.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_- DIMENSION_SET::COORDINATE_SYSTEM_DIMENSION_SET_PTR (`TYPE(COORDINATE_SYSTEM_TYPE),pointer` `COORDINATE_SYSTEM`, `INTEGER(INTG),intent(in)` `DIMENSION`, `INTEGER(INTG),intent(out)` `ERR`, `TYPE(VARYING_STRING),intent(out)` `ERROR`, `*`) [private]

Definition at line 1211 of file coordinate_routines.f90.

7.34 COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET Interface Reference

Private Member Functions

- subroutine [COORDINATE_SYSTEM_FOCUS_SET_NUMBER](#) (*USER_NUMBER*, *FOCUS*, *ERR*, *ERROR*,*)
- subroutine [COORDINATE_SYSTEM_FOCUS_SET_PTR](#) (*COORDINATE_SYSTEM*, *FOCUS*, *ERR*, *ERROR*,*)

7.34.1 Detailed Description

Definition at line 139 of file coordinate_routines.f90.

7.34.2 Member Function Documentation

7.34.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET::COORDINATE_SYSTEM_FOCUS_SET_NUMBER (*INTEGER(INTG),intent(in)* *USER_NUMBER*, *REAL(DP),intent(in)* *FOCUS*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING_STRING),intent(out)* *ERROR*, *) [private]

Definition at line 1281 of file coordinate_routines.f90.

7.34.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET::COORDINATE_SYSTEM_FOCUS_SET_PTR (*TYPE(COORDINATE_SYSTEM_TYPE),pointer* *COORDINATE_SYSTEM*, *REAL(DP),intent(in)* *FOCUS*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING_STRING),intent(out)* *ERROR*, *) [private]

Definition at line 1312 of file coordinate_routines.f90.

7.35 COORDINATE_ROUTINES::COORDINATE_SYSTEM_- ORIENTATION_SET Interface Reference

Private Member Functions

- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER` (`USER_NUMBER`, `ORIENTATION`, `ERR`, `ERROR,*`)
- subroutine `COORDINATE_SYSTEM_ORIENTATION_SET_PTR` (`COORDINATE_SYSTEM`, `ORIENTATION`, `ERR`, `ERROR,*`)

7.35.1 Detailed Description

Definition at line 159 of file coordinate_routines.f90.

7.35.2 Member Function Documentation

7.35.2.1 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
ORIENTATION_SET::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER`
(`INTEGER(INTG),intent(in) USER_NUMBER`, `REAL(DP),dimension(:, :, intent(in))
ORIENTATION`, `INTEGER(INTG),intent(out) ERR`, `TYPE(VARYING_-
STRING),intent(out) ERROR, *`) [private]

Definition at line 1624 of file coordinate_routines.f90.

7.35.2.2 subroutine `COORDINATE_ROUTINES::COORDINATE_SYSTEM_-
ORIENTATION_SET::COORDINATE_SYSTEM_ORIENTATION_SET_PTR`
(`TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,`
`REAL(DP),dimension(:, :, intent(in)) ORIENTATION`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Definition at line 1655 of file coordinate_routines.f90.

7.36 COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET Interface Reference

Private Member Functions

- subroutine [COORDINATE_SYSTEM_ORIGIN_SET_NUMBER](#) (*USER_NUMBER*, *ORIGIN*, *ERR*, *ERROR,**)
- subroutine [COORDINATE_SYSTEM_ORIGIN_SET_PTR](#) (*COORDINATE_SYSTEM*, *ORIGIN*, *ERR*, *ERROR,**)

7.36.1 Detailed Description

Definition at line 154 of file coordinate_routines.f90.

7.36.2 Member Function Documentation

7.36.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER
(*INTEGER(INTG),intent(in)* *USER_NUMBER*, *REAL(DP),dimension(:),intent(in)*
ORIGIN, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING_STRING),intent(out)*
*ERROR, *)* [private]

Definition at line 1552 of file coordinate_routines.f90.

7.36.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET::COORDINATE_SYSTEM_ORIGIN_SET_PTR
(*TYPE(COORDINATE_SYSTEM_TYPE),pointer* *COORDINATE_SYSTEM*,
REAL(DP),dimension(:),intent(in) *ORIGIN*, *INTEGER(INTG),intent(out)* *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR, *)* [private]

Definition at line 1583 of file coordinate_routines.f90.

7.37 COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE Struct Reference

Collaboration diagram for COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE:

Private Attributes

- TYPE(COORDINATE_SYSTEM_TYPE), pointer PTR

7.37.1 Detailed Description

Definition at line 95 of file coordinate_routines.f90.

7.37.2 Member Data Documentation

7.37.2.1 TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE::PTR [private]

Definition at line 96 of file coordinate_routines.f90.

7.38 COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET Interface

Reference **7.38 COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET Interface Reference** **751**

Private Member Functions

- subroutine [COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER](#) (USER_NUMBER, RADIAL_INTERPOLATION_TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR](#) (COORDINATE_SYSTEM, RADIAL_INTERPOLATION_TYPE, ERR, ERROR,*)

7.38.1 Detailed Description

Definition at line 144 of file coordinate_routines.f90.

7.38.2 Member Function Documentation

7.38.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(in) RADIAL_INTERPOLATION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1364 of file coordinate_routines.f90.

7.38.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR (TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM, INTEGER(INTG),intent(in) RADIAL_INTERPOLATION_TYPE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1395 of file coordinate_routines.f90.

7.39 COORDINATE_ROUTINES::COORDINATE_SYSTEM_- TYPE_SET Interface Reference

Private Member Functions

- subroutine [COORDINATE_SYSTEM_TYPE_SET_NUMBER](#) (USER_NUMBER, TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_SYSTEM_TYPE_SET_PTR](#) (COORDINATE_SYSTEM, TYPE, ERR, ERROR,*)

7.39.1 Detailed Description

Definition at line 149 of file coordinate_routines.f90.

7.39.2 Member Function Documentation

7.39.2.1 subroutine COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_-
SET::COORDINATE_SYSTEM_TYPE_SET_NUMBER (INTEGER(INTG),intent(in)
USER_NUMBER, INTEGER(INTG),intent(in) TYPE, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1471 of file coordinate_routines.f90.

7.39.2.2 subroutine COORDINATE_ROUTINES::COORDINATE_-
SYSTEM_TYPE_SET::COORDINATE_SYSTEM_TYPE_SET_PTR
(TYPE(COORDINATE_SYSTEM_TYPE),pointer COORDINATE_SYSTEM,
INTEGER(INTG),intent(in) TYPE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1502 of file coordinate_routines.f90.

7.40 COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE Struct Reference

Collaboration diagram for COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE:

Private Attributes

- INTEGER(INTG) [NUMBER_OF_COORDINATE_SYSTEMS](#)
- TYPE([COORDINATE_SYSTEM_PTR_TYPE](#)), pointer [COORDINATE_SYSTEMS](#)

7.40.1 Detailed Description

Definition at line 99 of file coordinate_routines.f90.

7.40.2 Member Data Documentation

7.40.2.1 TYPE([COORDINATE_SYSTEM_PTR_TYPE](#)),pointer [COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE::COORDINATE_SYSTEMS](#)
[private]

Definition at line 101 of file coordinate_routines.f90.

7.40.2.2 INTEGER(INTG) [COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE::NUMBER_OF_COORDINATE_SYSTEMS](#)
[private]

Definition at line 100 of file coordinate_routines.f90.

7.41 COORDINATE_ROUTINES::D2ZX Interface Reference

Private Member Functions

- REAL(DP) [D2ZX_DP](#) (COORDINATE_SYSTEM, I, J, X, ERR, ERROR)

7.41.1 Detailed Description

Definition at line 171 of file coordinate_routines.f90.

7.41.2 Member Function Documentation

- 7.41.2.1 REAL(DP) COORDINATE_ROUTINES::D2ZX::D2ZX_DP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) *COORDINATE_SYSTEM*, INTEGER(INTG),intent(in) *I*, INTEGER(INTG),intent(in) *J*, REAL(DP),dimension(:),intent(in) *X*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*)
[private]

Definition at line 2084 of file coordinate_routines.f90.

7.42 COORDINATE_ROUTINES::DXZ Interface Reference

Private Member Functions

- REAL(DP) [DXZ_DP](#) (COORDINATE_SYSTEM, I, X, ERR, ERROR)

7.42.1 Detailed Description

Definition at line 165 of file coordinate_routines.f90.

7.42.2 Member Function Documentation

7.42.2.1 **REAL(DP) COORDINATE_ROUTINES::DXZ::DXZ_DP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, INTEGER(INTG),intent(in) I, REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Definition at line 1943 of file coordinate_routines.f90.

7.43 COORDINATE_ROUTINES::DZX Interface Reference

Private Member Functions

- REAL(DP) [DZX_DP](#) (COORDINATE_SYSTEM, I, X, ERR, ERROR)

7.43.1 Detailed Description

Definition at line 177 of file coordinate_routines.f90.

7.43.2 Member Function Documentation

7.43.2.1 **REAL(DP) COORDINATE_ROUTINES::DZX::DZX_DP (TYPE(COORDINATE_SYSTEM_TYPE),intent(in) COORDINATE_SYSTEM, INTEGER(INTG),intent(in) I, REAL(DP),dimension(:),intent(in) X, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Definition at line 2353 of file coordinate_routines.f90.

7.44 EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF Interface Reference

Private Member Functions

- subroutine `EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS` (`EQUATIONS_SET`, `DOF_INDICES`, `CONDITIONS`, `VALUES`, `ERR`, `ERROR,*`)
- subroutine `EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1` (`EQUATIONS_SET`, `DOF_INDEX`, `CONDITION`, `VALUE`, `ERR`, `ERROR,*`)

7.44.1 Detailed Description

Definition at line 78 of file equations_set_routines.f90.

7.44.2 Member Function Documentation

7.44.2.1 subroutine `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1`
(`TYPE(EQUATIONS_SET_TYPE)`,pointer *EQUATIONS_SET*,
`INTEGER(INTG),intent(in) DOF_INDEX`, `INTEGER(INTG),intent(in) CONDITION`, `REAL(DP),intent(in) VALUE`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Parameters:

EQUATIONS_SET A pointer to the equations set to set the fixed condition for
DOF_INDEX The dof index to set the fixed condition at
CONDITION The fixed condition to set
VALUE The value of the fixed condition to set
ERR The error code
ERROR The error string

Definition at line 1678 of file equations_set_routines.f90.

7.44.2.2 subroutine `EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS` (`TYPE(EQUATIONS_SET_TYPE)`,pointer *EQUATIONS_SET*, `INTEGER(INTG),dimension(:),intent(in) DOF_INDICES`, `INTEGER(INTG),dimension(:,intent(in) CONDITIONS`,
`REAL(DP),dimension(:,intent(in) VALUES`, `INTEGER(INTG),intent(out) ERR`,
`TYPE(VARYING_STRING),intent(out) ERROR, *`) [private]

Parameters:

EQUATIONS_SET A pointer to the equations set to set the fixed conditions for.
DOF_INDICES `DOF_INDICES(i)`. The dof index for the i'th dof to set the fixed conditions for
CONDITIONS `CONDITIONS(i)`. The fixed condition for the i'th dof.
VALUES `VALUES(i)`. The value of the fixed condition for the i'th dof.
ERR The error code

ERROR The error string

Definition at line 1572 of file equations_set_routines.f90.

7.45 EQUATIONS_SET_ROUTINES::EQUATIONS_SET_- SPECIFICATION_SET Interface Reference

Private Member Functions

- EQUATIONS_SET_SPECIFICATION_SET_NUMBER
- EQUATIONS_SET_SPECIFICATION_SET_PTR

7.45.1 Detailed Description

Definition at line 83 of file equations_set_routines.f90.

7.45.2 Member Function Documentation

- 7.45.2.1 EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICATION_-
SET::EQUATIONS_SET_SPECIFICATION_SET_NUMBER ()
[private]
- 7.45.2.2 EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICATION_-
SET::EQUATIONS_SET_SPECIFICATION_SET_PTR () [private]

7.46 F90C::interface Interface Reference

Public Member Functions

- subroutine [CSTRINGLEN](#) (LENGTH, CSTRING)
- subroutine [PACKCHARACTERS](#) (INTCHAR, POSITION, CSTRING)
- subroutine [UNPACKCHARACTERS](#) (INTCHAR, POSITION, CSTRING)

7.46.1 Detailed Description

Definition at line 54 of file f90c_f.f90.

7.46.2 Member Function Documentation

7.46.2.1 subroutine F90C::interface::CSTRINGLEN (INTEGER(INTG),intent(out) LENGTH, INTEGER(INTG),dimension(*),intent(in) CSTRING)

Definition at line 56 of file f90c_f.f90.

7.46.2.2 subroutine F90C::interface::PACKCHARACTERS (INTE- GER(INTG),intent(in) INTCHAR, INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),dimension(*),intent(out) CSTRING)

Definition at line 63 of file f90c_f.f90.

7.46.2.3 subroutine F90C::interface::UNPACKCHARACTERS (INTE- GER(INTG),intent(out) INTCHAR, INTEGER(INTG),intent(in) POSITION, INTEGER(INTG),dimension(*),intent(in) CSTRING)

Definition at line 70 of file f90c_f.f90.

7.47 FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET:

Private Attributes

- TYPE(FIELDS_TYPE), pointer FIELDS
A pointer to the fields defined on the region.
- INTEGER(INTG) NUMBER_OF_ELEMENTS
Number of nodes in this computational node for NODAL_INFO_SET.
- INTEGER(INTG), allocatable LIST_OF_GLOBAL_NUMBER
the list of global numbering in each domain
- TYPE(FIELD_IO_INFO_SET), allocatable ELEMENTAL_INFO_SET
A list of nodal information for IO.

7.47.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 117 of file field_IO_routines.f90.

7.47.2 Member Data Documentation

7.47.2.1 TYPE(FIELD_IO_INFO_SET),allocatable FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET::ELEMENTAL_INFO_SET
[private]

A list of nodal information for IO.

Definition at line 122 of file field_IO_routines.f90.

7.47.2.2 TYPE(FIELDS_TYPE),pointer FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET::FIELDS [private]

A pointer to the fields defined on the region.

Definition at line 118 of file field_IO_routines.f90.

7.47.2.3 INTEGER(INTG),allocatable FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET::LIST_OF_GLOBAL_NUMBER
[private]

the list of global numbering in each domain

Definition at line 121 of file field_IO_routines.f90.

7.47.2.4 INTEGER(INTG) FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET::NUMBER_OF_ELEMENTS [private]

Number of nodes in this computational node for NODAL_INFO_SET.

Definition at line 119 of file field_IO_routines.f90.

7.48 FIELD_IO_ROUTINES::FIELD_IO_INFO_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD_IO_ROUTINES::FIELD_IO_INFO_SET:

Private Attributes

- LOGICAL [SAME_HEADER](#)
determine whether we have same IO information as the previous one
- INTEGER(INTG) [NUMBER_OF_COMPONENTS](#)
number of components in the component array, COMPONENT(:)
- TYPE([FIELD_VARIABLE_COMPONENT_PTR_TYPE](#)), allocatable [COMPONENTS](#)
A array of pointers to those components of the node in this local domain.

7.48.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 98 of file field_IO_routines.f90.

7.48.2 Member Data Documentation

7.48.2.1 TYPE(FIELD_VARIABLE_COMPONENT_PTR_TYPE),allocatable FIELD_IO_ROUTINES::FIELD_IO_INFO_SET::COMPONENTS [private]

A array of pointers to those components of the node in this local domain.

Definition at line 104 of file field_IO_routines.f90.

7.48.2.2 INTEGER(INTG) FIELD_IO_ROUTINES::FIELD_IO_INFO_SET::NUMBER_OF_- COMPONENTS [private]

number of components in the component array, COMPONENT(:)

Definition at line 101 of file field_IO_routines.f90.

7.48.2.3 LOGICAL FIELD_IO_ROUTINES::FIELD_IO_INFO_SET::SAME_HEADER [private]

determine whether we have same IO information as the previous one

Definition at line 100 of file field_IO_routines.f90.

7.49 FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET Struct Reference

contains information for parallel IO, and it is nodal base

Collaboration diagram for FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET:

Private Attributes

- TYPE(FIELDS_TYPE), pointer FIELDS
A pointe r to the fields defined on the region.
- INTEGER(INTG) NUMBER_OF_NODES
Number of nodes in this computational node for NODAL_INFO_SET.
- INTEGER(INTG), allocatable LIST_OF_GLOBAL_NUMBER
the list of global numbering in each domain
- TYPE(FIELD_IO_INFO_SET), allocatable NODAL_INFO_SET
A list of nodal information for IO.

7.49.1 Detailed Description

contains information for parallel IO, and it is nodal base

Definition at line 108 of file field_IO_routines.f90.

7.49.2 Member Data Documentation

7.49.2.1 TYPE(FIELDS_TYPE),pointer FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET::FIELDS [private]

A pointe r to the fields defined on the region.

Definition at line 109 of file field_IO_routines.f90.

7.49.2.2 INTEGER(INTG),allocatable FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET::LIST_OF_GLOBAL_NUMBER [private]

the list of global numbering in each domain

Definition at line 112 of file field_IO_routines.f90.

7.49.2.3 TYPE(FIELD_IO_INFO_SET),allocatable FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET::NODAL_INFO_SET [private]

A list of nodal information for IO.

Definition at line 113 of file field_IO_routines.f90.

7.49.2.4 INTEGER(INTG) FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET::NUMBER_OF_NODES [private]

Number of nodes in this computational node for NODAL_INFO_SET.

Definition at line 110 of file field_IO_routines.f90.

7.50 FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE Struct Reference

field variable component type pointer for IO

Collaboration diagram for FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE:

Private Attributes

- TYPE(FIELD_VARIABLE_COMPONENT_TYPE), pointer PTR
pointer field variable component

7.50.1 Detailed Description

field variable component type pointer for IO

Definition at line 93 of file field_IO_routines.f90.

7.50.2 Member Data Documentation

7.50.2.1 TYPE(FIELD_VARIABLE_COMPONENT_TYPE),pointer FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE::PTR [private]

pointer field variable component

Definition at line 94 of file field_IO_routines.f90.

7.51 FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE Struct Reference

field variable component type pointer for IO

Collaboration diagram for FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE:

Private Attributes

- TYPE(MESH_ELEMENTS_TYPE), pointer PTR
pointer field variable component

7.51.1 Detailed Description

field variable component type pointer for IO

Definition at line 88 of file field_IO_routines.f90.

7.51.2 Member Data Documentation

7.51.2.1 TYPE(MESH_ELEMENTS_TYPE),pointer FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE::PTR [private]

pointer field variable component

Definition at line 89 of file field_IO_routines.f90.

7.52 FIELD_ROUTINES::FIELD_COMPONENT_- INTERPOLATION_SET Interface Reference

Private Member Functions

- FIELD_COMPONENT_INTERPOLATION_SET_NUMBER
- FIELD_COMPONENT_INTERPOLATION_SET_PTR

7.52.1 Detailed Description

Definition at line 152 of file field_routines.f90.

7.52.2 Member Function Documentation

7.52.2.1 FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_-
SET::FIELD_COMPONENT_INTERPOLATION_SET_NUMBER ()
[private]

7.52.2.2 FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_-
SET::FIELD_COMPONENT_INTERPOLATION_SET_PTR ()
[private]

7.53 FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET Interface Reference

Private Member Functions

- [FIELD_COMPONENT_MESH_COMPONENT_SET_NUMBER](#)
- [FIELD_COMPONENT_MESH_COMPONENT_SET_PTR](#)

7.53.1 Detailed Description

Definition at line 157 of file field_routines.f90.

7.53.2 Member Function Documentation

7.53.2.1 FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET::FIELD_COMPONENT_MESH_COMPONENT_SET_NUMBER ()
[private]

7.53.2.2 FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET::FIELD_COMPONENT_MESH_COMPONENT_SET_PTR ()
[private]

7.54 FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET Interface Reference

Private Member Functions

- subroutine [FIELD_DEPENDENT_TYPE_SET_NUMBER](#) (*USER_NUMBER*, *REGION*, *DEPENDENT_TYPE*, *ERR*, *ERROR,**)
- subroutine [FIELD_DEPENDENT_TYPE_SET_PTR](#) (*FIELD*, *DEPENDENT_TYPE*, *ERR*, *ERROR,**)

7.54.1 Detailed Description

Definition at line 162 of file field_routines.f90.

7.54.2 Member Function Documentation

7.54.2.1 subroutine FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET::FIELD_DEPENDENT_TYPE_SET_NUMBER (*INTEGER(INTG),intent(in)* *USER_NUMBER*, *TYPE(REGION_TYPE),pointer REGION*, *INTEGER(INTG),intent(in)* *DEPENDENT_TYPE*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING_STRING),intent(out)* *ERROR, **) [private]

Parameters:

USER_NUMBER The user number of the field to set the dependent type for
REGION A pointer to the region containing the field

DEPENDENT_TYPE The dependent type to set/change for the field

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1015 of file field_routines.f90.

7.54.2.2 subroutine FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET::FIELD_DEPENDENT_TYPE_SET_PTR (*TYPE(FIELD_TYPE),pointer FIELD*, *INTEGER(INTG),intent(in)* *DEPENDENT_TYPE*, *INTEGER(INTG),intent(out)* *ERR*, *TYPE(VARYING_STRING),intent(out)* *ERROR, **) [private]

Parameters:

FIELD A pointer to the field to set/change the dependent type for

DEPENDENT_TYPE The dependent type to set/change

See also:

[FIELD_ROUTINES::DependentTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1046 of file field_routines.f90.

7.55 FIELD_ROUTINES::FIELD_DIMENSION_SET Interface Reference

Private Member Functions

- subroutine [FIELD_DIMENSION_SET_NUMBER](#) (USER_NUMBER, REGION, FIELD_DIMENSION, ERR, ERROR,*)
- subroutine [FIELD_DIMENSION_SET_PTR](#) (FIELD, FIELD_DIMENSION, ERR, ERROR,*)

7.55.1 Detailed Description

Definition at line 167 of file field_routines.f90.

7.55.2 Member Function Documentation

7.55.2.1 subroutine FIELD_ROUTINES::FIELD_DIMENSION_SET::FIELD_DIMENSION_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(REGION_TYPE),pointer *REGION*, INTEGER(INTG),intent(in) *FIELD_DIMENSION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

USER_NUMBER The user number of the field to set the dimension for

REGION A pointer to the region containing the field

FIELD_DIMENSION The dimension to set/change

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1219 of file field_routines.f90.

7.55.2.2 subroutine FIELD_ROUTINES::FIELD_DIMENSION_SET::FIELD_DIMENSION_SET_PTR (TYPE(FIELD_TYPE),pointer *FIELD*, INTEGER(INTG),intent(in) *FIELD_DIMENSION*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

FIELD A pointer to the field to set/change the dimension for

FIELD_DIMENSION The field dimension to set/change

See also:

[FIELD_ROUTINES::DimensionTypes](#),[FIELD_ROUTINES](#)

ERR The error code

ERROR The error string

Definition at line 1250 of file field_routines.f90.

7.56 FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET Interface Reference

Private Member Functions

- [FIELD_GEOMETRIC_FIELD_SET_NUMBER](#)
- [FIELD_GEOMETRIC_FIELD_SET_PTR](#)

7.56.1 Detailed Description

Definition at line 172 of file field_routines.f90.

7.56.2 Member Function Documentation

7.56.2.1 FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET::FIELD_GEOMETRIC_FIELD_SET_NUMBER () [private]

7.56.2.2 FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET::FIELD_GEOMETRIC_FIELD_SET_PTR () [private]

7.57 FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET Interface Reference

Private Member Functions

- [FIELD_MESH_DECOMPOSITION_SET_NUMBER](#)
- [FIELD_MESH_DECOMPOSITION_SET_PTR](#)

7.57.1 Detailed Description

Definition at line 177 of file field_routines.f90.

7.57.2 Member Function Documentation

7.57.2.1 FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET::FIELD_MESH_DECOMPOSITION_SET_NUMBER () [private]

7.57.2.2 FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET::FIELD_MESH_DECOMPOSITION_SET_PTR () [private]

7.58 FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET Interface Reference

Private Member Functions

- [FIELD_NUMBER_OF_COMPONENTS_SET_NUMBER](#)
- [FIELD_NUMBER_OF_COMPONENTS_SET_PTR](#)

7.58.1 Detailed Description

Definition at line 182 of file field_routines.f90.

7.58.2 Member Function Documentation

7.58.2.1 FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET::FIELD_NUMBER_OF_COMPONENTS_SET_NUMBER () [private]

7.58.2.2 FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET::FIELD_NUMBER_OF_COMPONENTS_SET_PTR () [private]

7.59 FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET Interface Reference

Private Member Functions

- [FIELD_NUMBER_OF_VARIABLES_SET_NUMBER](#)
- [FIELD_NUMBER_OF_VARIABLES_SET_PTR](#)

7.59.1 Detailed Description

Definition at line 187 of file field_routines.f90.

7.59.2 Member Function Documentation

7.59.2.1 FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET::FIELD_NUMBER_OF_VARIABLES_SET_NUMBER () [private]

7.59.2.2 FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET::FIELD_NUMBER_OF_VARIABLES_SET_PTR () [private]

7.60 FIELD_ROUTINES::FIELD_SCALING_TYPE_SET Interface Reference

Private Member Functions

- FIELD_SCALING_TYPE_SET_NUMBER
- FIELD_SCALING_TYPE_SET_PTR

7.60.1 Detailed Description

Definition at line 192 of file field_routines.f90.

7.60.2 Member Function Documentation

7.60.2.1 **FIELD_ROUTINES::FIELD_SCALING_TYPE_SET::FIELD_SCALING_TYPE_SET_-
NUMBER () [private]**

7.60.2.2 **FIELD_ROUTINES::FIELD_SCALING_TYPE_SET::FIELD_SCALING_TYPE_SET_-
PTR () [private]**

7.61 FIELD_ROUTINES::FIELD_TYPE_SET Interface Reference

Private Member Functions

- [FIELD_TYPE_SET_NUMBER](#)
- [FIELD_TYPE_SET_PTR](#)

7.61.1 Detailed Description

Definition at line 197 of file field_routines.f90.

7.61.2 Member Function Documentation

7.61.2.1 FIELD_ROUTINES::FIELD_TYPE_SET::FIELD_TYPE_SET_NUMBER ()
[private]

7.61.2.2 FIELD_ROUTINES::FIELD_TYPE_SET::FIELD_TYPE_SET_PTR () [private]

7.62 INPUT_OUTPUT::WRITE_STRING Interface Reference

Write a string to a given output stream.

Private Member Functions

- subroutine [WRITE_STRING_C](#) (ID, STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_VS](#) (ID, STRING, ERR, ERROR,*)

7.62.1 Detailed Description

Write a string to a given output stream.

Definition at line 71 of file input_output.f90.

7.62.2 Member Function Documentation

7.62.2.1 subroutine INPUT_OUTPUT::WRITE_STRING::WRITE_STRING_C
`(INTEGER(INTG),intent(in) ID, CHARACTER(LEN=*),intent(in) STRING,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
 [private]

Parameters:

ID The ID of the output stream to write to

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

STRING The string to write

ERR The error code

ERROR The error string

Definition at line 221 of file input_output.f90.

7.62.2.2 subroutine INPUT_OUTPUT::WRITE_STRING::WRITE_STRING_VS
`(INTEGER(INTG),intent(in) ID, TYPE(VARYING_STRING),intent(in) STRING,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)`
 [private]

Parameters:

ID The ID of the output stream to write to

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

STRING The string to write

ERR The error code

ERROR The error string

Definition at line 247 of file input_output.f90.

7.63 INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE Interface Reference

Write a string, value, string then a value with the values formatted in a particular way to a given output stream.

Private Member Functions

- [WRITE_STRING_FMT_TWO_VALUE_C_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_C_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_C_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_C_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_C_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_C_VS](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_DP_VS](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_INTG_VS](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_L_VS](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_SP_VS](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_C](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_DP](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_INTG](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_L](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_SP](#)
- [WRITE_STRING_FMT_TWO_VALUE_VS_VS](#)

7.63.1 Detailed Description

Write a string, value, string then a value with the values formatted in a particular way to a given output stream.

Definition at line 139 of file input_output.f90.

7.63.2 Member Function Documentation

- 7.63.2.1 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_C()` [private]
- 7.63.2.2 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_DP()` [private]
- 7.63.2.3 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_INTG()` [private]
- 7.63.2.4 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_L()` [private]
- 7.63.2.5 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_SP()` [private]
- 7.63.2.6 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_C_VS()` [private]
- 7.63.2.7 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_C()` [private]
- 7.63.2.8 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_DP()` [private]
- 7.63.2.9 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_INTG()` [private]
- 7.63.2.10 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_L()` [private]
- 7.63.2.11 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_SP()` [private]
- 7.63.2.12 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_DP_VS()` [private]
- 7.63.2.13 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_C()` [private]
- 7.63.2.14 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_DP()` [private]
- 7.63.2.15 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_INTG()` [private]
- 7.63.2.16 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_L()` [private]
- 7.63.2.17 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_SP()` [private]
- 7.63.2.18 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_INTG_VS()` [private]
- 7.63.2.19 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_L_C()` [private]
- 7.63.2.20 `INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE::WRITE_STRING_FMT_-
TWO_VALUE_L_DP()` [private]

7.64 INPUT_OUTPUT::WRITE_STRING_FMT_VALUE Interface Reference

Write a string followed by a value formatted in a particular way to a specified output stream.

Private Member Functions

- subroutine [WRITE_STRING_FMT_VALUE_C](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_DP](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_INTG](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_LINTG](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_L](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_SP](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)
- subroutine [WRITE_STRING_FMT_VALUE_VS](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

7.64.1 Detailed Description

Write a string followed by a value formatted in a particular way to a specified output stream.

Definition at line 128 of file input_output.f90.

7.64.2 Member Function Documentation

7.64.2.1 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_STRING_FMT_VALUE_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *VALUE*, CHARACTER(LEN=*),intent(in) *FORMAT_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1680 of file input_output.f90.

7.64.2.2 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_STRING_FMT_VALUE_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *VALUE*, CHARACTER(LEN=*),intent(in) *FORMAT_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1710 of file input_output.f90.

7.64.2.3 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_STRING_FMT_VALUE_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *VALUE*, CHARACTER(LEN=*),intent(in) *FORMAT_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1741 of file input_output.f90.

7.64.2.4 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_STRING_FMT_VALUE_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *VALUE*, CHARACTER(LEN=*),intent(in) *FORMAT_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1801 of file input_output.f90.

```
7.64.2.5 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_-
STRING_FMT_VALUE_LINTG (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, INTEGER(LINTG),intent(in)
VALUE, CHARACTER(LEN=*),intent(in) FORMAT_STRING,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)
[private]
```

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1771 of file input_output.f90.

```
7.64.2.6 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_-
STRING_FMT_VALUE_SP (INTEGER(INTG),intent(in) ID,
CHARACTER(LEN=*),intent(in) FIRST_STRING, REAL(SP),intent(in) VALUE,
CHARACTER(LEN=*),intent(in) FORMAT_STRING, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1831 of file input_output.f90.

7.64.2.7 subroutine INPUT_OUTPUT::WRITE_STRING_FMT_VALUE::WRITE_STRING_FMT_VALUE_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *VALUE*, CHARACTER(LEN=*),intent(in) *FORMAT_STRING*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

FORMAT_STRING The format string to be used to format the value

ERR The error code

ERROR The error string

Definition at line 1862 of file input_output.f90.

7.65 INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR Interface Reference

Write a string followed by a indexed vector to a specified output stream.

Private Member Functions

- [WRITE_STRING_IDX_VECTOR_DP](#)
- [WRITE_STRING_IDX_VECTOR_INTG](#)
- [WRITE_STRING_IDX_VECTOR_LINTG](#)
- [WRITE_STRING_IDX_VECTOR_L](#)
- [WRITE_STRING_IDX_VECTOR_SP](#)

7.65.1 Detailed Description

Write a string followed by a indexed vector to a specified output stream.

Definition at line 188 of file input_output.f90.

7.65.2 Member Function Documentation

- 7.65.2.1 [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR::WRITE_STRING_IDX_VECTOR_DP \(\)](#) [private]
- 7.65.2.2 [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR::WRITE_STRING_IDX_VECTOR_INTG \(\)](#) [private]
- 7.65.2.3 [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR::WRITE_STRING_IDX_VECTOR_L \(\)](#) [private]
- 7.65.2.4 [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR::WRITE_STRING_IDX_VECTOR_LINTG \(\)](#) [private]
- 7.65.2.5 [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR::WRITE_STRING_IDX_VECTOR_SP \(\)](#) [private]

7.66 INPUT_OUTPUT::WRITE_STRING_MATRIX Interface Reference

Write a string followed by a matrix to a specified output stream.

Private Member Functions

- subroutine `WRITE_STRING_MATRIX_DP` (*ID*, *FIRST_ROW*, *DELTA_ROW*, *LAST_ROW*, *FIRST_COLUMN*, *DELTA_COLUMN*, *LAST_COLUMN*, *NUMBER_FIRS*,&*NUMBER_REPEAT*, *MATRIX*, *INDEX_FORMAT_TYPE*, *MATRIX_NAME_FORMAT*, *ROW_INDEX_FORMAT*, *FIRST_FORMAT*, *REPEAT_FORMAT*, *ERR*, *ERROR*,*)
- subroutine `WRITE_STRING_MATRIX_INTG` (*ID*, *FIRST_ROW*, *DELTA_ROW*, *LAST_ROW*, *FIRST_COLUMN*, *DELTA_COLUMN*, *LAST_COLUMN*, *NUMBER_FI ST*,&*NUMBER_REPEAT*, *MATRIX*, *INDEX_FORMAT_TYPE*, *MATRIX_NAME_FORMAT*, *ROW_INDEX_FORMAT*, *FIRST_FORMAT*, *REPEAT_FORMAT*, *ERR*, *ERROR*,*)
- subroutine `WRITE_STRING_MATRIX_LINTG` (*ID*, *FIRST_ROW*, *DELTA_ROW*, *LAST_ROW*, *FIRST_COLUMN*, *DELTA_COLUMN*, *LAST_COLUMN*, *NUMBER_F RST*,&*NUMBER_REPEAT*, *MATRIX*, *INDEX_FORMAT_TYPE*, *MATRIX_NAME_FORMAT*, *ROW_INDEX_FORMAT*, *FIRST_FORMAT*, *REPEAT_FORMAT*, *ERR*, *ERROR*,*)
- subroutine `WRITE_STRING_MATRIX_L` (*ID*, *FIRST_ROW*, *DELTA_ROW*, *LAST_ROW*, *FIRST_COLUMN*, *DELTA_COLUMN*, *LAST_COLUMN*, *NUMBER_FIRST* &*NUMBER_REPEAT*, *MATRIX*, *INDEX_FORMAT_TYPE*, *MATRIX_NAME_FORMAT*, *ROW_INDEX_FORMAT*, *FIRST_FORMAT*, *REPEAT_FORMAT*, *ERR*, *ERROR*,*)
- subroutine `WRITE_STRING_MATRIX_SP` (*ID*, *FIRST_ROW*, *DELTA_ROW*, *LAST_ROW*, *FIRST_COLUMN*, *DELTA_COLUMN*, *LAST_COLUMN*, *NUMBER_FIRS*,&*NUMBER_REPEAT*, *MATRIX*, *INDEX_FORMAT_TYPE*, *MATRIX_NAME_FORMAT*, *ROW_INDEX_FORMAT*, *FIRST_FORMAT*, *REPEAT_FORMAT*, *ERR*, *ERROR*,*)

7.66.1 Detailed Description

Write a string followed by a matrix to a specified output stream.

Definition at line 197 of file input_output.f90.

7.66.2 Member Function Documentation

**7.66.2.1 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX::WRITE_STRING_-
MATRIX_DP (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)
FIRST_ROW, INTEGER(INTG),intent(in) *DELTA_ROW*, INTEGER(INTG),intent(in)
LAST_ROW, INTEGER(INTG),intent(in) *FIRST_COLUMN*,
INTEGER(INTG),intent(in) *DELTA_COLUMN*, INTEGER(INTG),intent(in)
LAST_COLUMN, *NUMBER_FIRS*, &,intent(in) *NUMBER_REPEAT*,
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)
INDEX_FORMAT_TYPE, CHARACTER(LEN=*=),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*=),intent(in)
ROW_INDEX_FORMAT, CHARACTER(LEN=*=),intent(in) *FIRST_FORMAT*,
CHARACTER(LEN=*=),intent(in) *REPEAT_FORMAT*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#),[INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3655 of file input_output.f90.

**7.66.2.2 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX::WRITE_STRING_-
MATRIX_INTG (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)
FIRST_ROW, INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
LAST_COLUMN, NUMBER_FI ST, &,intent(in) NUMBER_REPEAT,
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)
INDEX_FORMAT_TYPE, CHARACTER(LEN=*),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*),intent(in)
ROW_INDEX_FORMAT, CHARACTER(LEN=*),intent(in) FIRST_FORMAT,
CHARACTER(LEN=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputing the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#),[INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3720 of file input_output.f90.

**7.66.2.3 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX::WRITE_STRING_-
MATRIX_L (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in) *FIRST_ROW*,
INTEGER(INTG),intent(in) *DELTA_ROW*, INTEGER(INTG),intent(in) *LAST_ROW*,
INTEGER(INTG),intent(in) *FIRST_COLUMN*, INTEGER(INTG),intent(in)
DELTA_COLUMN, INTEGER(INTG),intent(in) *LAST_COLUMN*, NUMBER_FIRST
&,intent(in) *NUMBER_REPEAT*, INTEGER(INTG),dimension(:, :, :),intent(in) *matrix*,
INTEGER(INTG),intent(in) *INDEX_FORMAT_TYPE*, CHARACTER(LEN=*=*),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in) *ROW_INDEX_-
FORMAT*, CHARACTER(LEN=*=*),intent(in) *FIRST_FORMAT*,
CHARACTER(LEN=*=*),intent(in) *REPEAT_FORMAT*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputing the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputing the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#),[INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3857 of file input_output.f90.

```
7.66.2.4 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX::WRITE_STRING_-
    MATRIX_LINTG (INTEGER(INTG),intent(in) ID, INTEGER(INTG),intent(in)
    FIRST_ROW, INTEGER(INTG),intent(in) DELTA_ROW, INTEGER(INTG),intent(in)
    LAST_ROW, INTEGER(INTG),intent(in) FIRST_COLUMN,
    INTEGER(INTG),intent(in) DELTA_COLUMN, INTEGER(INTG),intent(in)
    LAST_COLUMN, NUMBER_F RST, &,intent(in) NUMBER_REPEAT,
    INTEGER(INTG),dimension(:, :, intent(in) matrix, INTEGER(INTG),intent(in)
    INDEX_FORMAT_TYPE, CHARACTER(LEN=*=*),intent(in)
    MATRIX_NAME_FORMAT, CHARACTER(LEN=*=*),intent(in)
    ROW_INDEX_FORMAT, CHARACTER(LEN=*=*),intent(in) FIRST_FORMAT,
    CHARACTER(LEN=*=*),intent(in) REPEAT_FORMAT, INTEGER(INTG),intent(out)
    ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]
```

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputing the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputing the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3792 of file input_output.f90.

**7.66.2.5 subroutine INPUT_OUTPUT::WRITE_STRING_MATRIX::WRITE_STRING_-
MATRIX_SP (INTEGER(INTG),intent(in) *ID*, INTEGER(INTG),intent(in)
FIRST_ROW, INTEGER(INTG),intent(in) *DELTA_ROW*, INTEGER(INTG),intent(in)
LAST_ROW, INTEGER(INTG),intent(in) *FIRST_COLUMN*,
INTEGER(INTG),intent(in) *DELTA_COLUMN*, INTEGER(INTG),intent(in)
LAST_COLUMN, NUMBER_FIRS, &,intent(in) *NUMBER_REPEAT*,
INTEGER(INTG),dimension(:, :, intent(in) *matrix*, INTEGER(INTG),intent(in)
INDEX_FORMAT_TYPE, CHARACTER(LEN=*=),intent(in)
MATRIX_NAME_FORMAT, CHARACTER(LEN=*=),intent(in)
ROW_INDEX_FORMAT, CHARACTER(LEN=*=),intent(in) *FIRST_FORMAT*,
CHARACTER(LEN=*=),intent(in) *REPEAT_FORMAT*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_ROW The first row of the matrix to be output

DELTA_ROW The delta row increment to be used when outputting the first through to the last matrix row

LAST_ROW The last row of the matrix to be output

FIRST_COLUMN The first column of the matrix to be output

DELTA_COLUMN The delta column increase to be used when outputting the first through to the last matrix column

LAST_COLUMN The last column of the matrix to be output

INDEX_FORMAT_TYPE The format type to be used for the matrix name and indices

See also:

[INPUT_OUTPUT::MatrixNameIndexFormat](#), [INPUT_OUTPUT::MatrixNameIndexFormat](#)

MATRIX_NAME_FORMAT The format string to be used to format the matrix name

ROW_INDEX_FORMAT The format string to be used to format the row indices

FIRST_FORMAT The format string to be used for the first line of output

REPEAT_FORMAT The format type to be used for the second and subsequently repeated lines of output

ERR The error code

ERROR The error string

Definition at line 3922 of file input_output.f90.

7.67 INPUT_OUTPUT::WRITE_STRING_TWO_VALUE Interface Reference

Write a string, value, string then a value to a given output stream.

Private Member Functions

- subroutine `WRITE_STRING_TWO_VALUE_C_C` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_C_DP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_C_INTG` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_C_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_C_SP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_C_VS` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_C` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_DP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_INTG` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_SP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_DP_VS` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_C` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_DP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_INTG` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_SP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_INTG_VS` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_L_C` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_L_DP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_L_INTG` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)
- subroutine `WRITE_STRING_TWO_VALUE_L_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

- subroutine `WRITE_STRING_TWO_VALUE_L_SP` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_L_VS` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_C` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_DP` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_INTG` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_L` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_SP` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_SP_VS` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_C` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_DP` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_INTG` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_L` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_SP` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)
- subroutine `WRITE_STRING_TWO_VALUE_VS_VS` (*ID*, *FIRST_STRING*, *FIRST_VALUE*, *SECOND_STRING*, *SECOND_VALUE*, *ERR*, *ERROR,**)

7.67.1 Detailed Description

Write a string, value, string then a value to a given output stream.

Definition at line 88 of file input_output.f90.

7.67.2 Member Function Documentation

7.67.2.1 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

`BASE_ROUTINES::OutputType`, `BASE_ROUTINES::FileUnits`

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output
SECOND_VALUE The second value to be output
ERR The error code
ERROR The error string

Definition at line 480 of file input_output.f90.

7.67.2.2 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_DP (INTEGER(INTG),intent(in) **ID**, CHARACTER(LEN=*),intent(in) **FIRST_STRING**, CHARACTER(LEN=*),intent(in) **FIRST_VALUE**, CHARACTER(LEN=*),intent(in) **SECOND_STRING**, REAL(DP),intent(in) **SECOND_VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 511 of file input_output.f90.

7.67.2.3 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_INTG (INTEGER(INTG),intent(in) **ID**, CHARACTER(LEN=*),intent(in) **FIRST_STRING**, CHARACTER(LEN=*),intent(in) **FIRST_VALUE**, CHARACTER(LEN=*),intent(in) **SECOND_STRING**, INTEGER(INTG),intent(in) **SECOND_VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 543 of file input_output.f90.

7.67.2.4 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 575 of file input_output.f90.

7.67.2.5 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 606 of file input_output.f90.

7.67.2.6 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_C_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 638 of file input_output.f90.

7.67.2.7 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 669 of file input_output.f90.

7.67.2.8 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 701 of file input_output.f90.

7.67.2.9 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 737 of file input_output.f90.

7.67.2.10 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 773 of file input_output.f90.

7.67.2.11 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 808 of file input_output.f90.

7.67.2.12 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_DP_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 844 of file input_output.f90.

7.67.2.13 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 876 of file input_output.f90.

7.67.2.14 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 908 of file input_output.f90.

7.67.2.15 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 944 of file input_output.f90.

7.67.2.16 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 980 of file input_output.f90.

7.67.2.17 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1015 of file input_output.f90.

7.67.2.18 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_INTG_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, INTEGER(INTG),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1051 of file input_output.f90.

7.67.2.19 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1083 of file input_output.f90.

7.67.2.20 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1115 of file input_output.f90.

7.67.2.21 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1151 of file input_output.f90.

7.67.2.22 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1185 of file input_output.f90.

7.67.2.23 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1220 of file input_output.f90.

7.67.2.24 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_L_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, LOGICAL,intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1254 of file input_output.f90.

7.67.2.25 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1286 of file input_output.f90.

7.67.2.26 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1318 of file input_output.f90.

7.67.2.27 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1354 of file input_output.f90.

7.67.2.28 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1390 of file input_output.f90.

7.67.2.29 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1423 of file input_output.f90.

7.67.2.30 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_SP_VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(SP),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, TYPE(VARYING_STRING),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1459 of file input_output.f90.

7.67.2.31 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_VS_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, CHARACTER(LEN=*),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1491 of file input_output.f90.

7.67.2.32 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_VS_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(DP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1522 of file input_output.f90.

7.67.2.33 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_VS_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, INTEGER(INTG),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1554 of file input_output.f90.

7.67.2.34 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_VS_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, LOGICAL,intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1586 of file input_output.f90.

7.67.2.35 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_STRING_TWO_VALUE_VS_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, TYPE(VARYING_STRING),intent(in) *FIRST_VALUE*, CHARACTER(LEN=*),intent(in) *SECOND_STRING*, REAL(SP),intent(in) *SECOND_VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1617 of file input_output.f90.

```
7.67.2.36 subroutine INPUT_OUTPUT::WRITE_STRING_TWO_VALUE::WRITE_-
    STRING_TWO_VALUE_VS_VS (INTEGER(INTG),intent(in) ID,
    CHARACTER(LEN=*),intent(in) FIRST_STRING, TYPE(VARYING_-
    STRING),intent(in) FIRST_VALUE, CHARACTER(LEN=*),intent(in)
    SECOND_STRING, TYPE(VARYING_STRING),intent(in) SECOND_VALUE,
    INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
    *) [private]
```

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

FIRST_VALUE The first value to be output

SECOND_STRING The second string to be output

SECOND_VALUE The second value to be output

ERR The error code

ERROR The error string

Definition at line 1649 of file input_output.f90.

7.68 INPUT_OUTPUT::WRITE_STRING_VALUE Interface Reference

Write a string followed by a value to a given output stream.

Private Member Functions

- subroutine [WRITE_STRING_VALUE_C](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_DP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_INTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_LINTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_L](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_SP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)
- subroutine [WRITE_STRING_VALUE_VS](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

7.68.1 Detailed Description

Write a string followed by a value to a given output stream.

Definition at line 77 of file input_output.f90.

7.68.2 Member Function Documentation

7.68.2.1 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_VALUE_C (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, CHARACTER(LEN=*),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#),[BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 273 of file input_output.f90.

7.68.2.2 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_VALUE_DP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 302 of file input_output.f90.

**7.68.2.3 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_-
VALUE_INTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in)
FIRST_STRING, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 332 of file input_output.f90.

**7.68.2.4 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_-
VALUE_L (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in)
FIRST_STRING, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 392 of file input_output.f90.

**7.68.2.5 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_-
VALUE_LINTG (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in)
FIRST_STRING, INTEGER(LINTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 362 of file input_output.f90.

**7.68.2.6 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_-
VALUE_SP (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in)
FIRST_STRING, REAL(SP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 421 of file input_output.f90.

**7.68.2.7 subroutine INPUT_OUTPUT::WRITE_STRING_VALUE::WRITE_STRING_VALUE_-
VS (INTEGER(INTG),intent(in) *ID*, CHARACTER(LEN=*),intent(in) *FIRST_STRING*,
TYPE(VARYING_STRING),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]**

Parameters:

ID The ID of the output stream. An ID of > 9 specifies file output

See also:

[BASE_ROUTINES::OutputType](#), [BASE_ROUTINES::FileUnits](#)

FIRST_STRING The first string to be output

VALUE The value to be output

ERR The error code

ERROR The error string

Definition at line 451 of file input_output.f90.

7.69 INPUT_OUTPUT::WRITE_STRING_VECTOR Interface Reference

Write a string followed by a vector to a specified output stream.

Private Member Functions

- [WRITE_STRING_VECTOR_DP](#)
- [WRITE_STRING_VECTOR_INTG](#)
- [WRITE_STRING_VECTOR_LINTG](#)
- [WRITE_STRING_VECTOR_L](#)
- [WRITE_STRING_VECTOR_SP](#)

7.69.1 Detailed Description

Write a string followed by a vector to a specified output stream.

Definition at line 179 of file input_output.f90.

7.69.2 Member Function Documentation

7.69.2.1 INPUT_OUTPUT::WRITE_STRING_VECTOR::WRITE_STRING_VECTOR_DP ()
[private]

7.69.2.2 INPUT_OUTPUT::WRITE_STRING_VECTOR::WRITE_STRING_VECTOR_INTG ()
[private]

7.69.2.3 INPUT_OUTPUT::WRITE_STRING_VECTOR::WRITE_STRING_VECTOR_L ()
[private]

7.69.2.4 INPUT_OUTPUT::WRITE_STRING_VECTOR::WRITE_STRING_VECTOR_LINTG ()
[private]

7.69.2.5 INPUT_OUTPUT::WRITE_STRING_VECTOR::WRITE_STRING_VECTOR_SP ()
[private]

7.70 ISO_VARYING_STRING::adjustr Interface Reference

Public Member Functions

- type(varying_string) **adjustr_** (string)

7.70.1 Detailed Description

Definition at line 116 of file iso_varying_string.f90.

7.70.2 Member Function Documentation

7.70.2.1 type(varying_string) ISO_VARYING_STRING::adjustr::adjustr_(type(varying_string),intent(in) string)

Definition at line 875 of file iso_varying_string.f90.

7.71 ISO_VARYING_STRING::assignment(=) Interface Reference

Public Member Functions

- subroutine `op_assign_CH_VS` (var, exp)
- subroutine `op_assign_VS_CH` (var, exp)
- type(varying_string) `op_concat_VS_VS` (string_a, string_b)
- type(varying_string) `op_concat_CH_VS` (string_a, string_b)
- type(varying_string) `op_concat_VS_CH` (string_a, string_b)
- logical `op_eq_VS_VS` (string_a, string_b)
- logical `op_eq_CH_VS` (string_a, string_b)
- logical `op_eq_VS_CH` (string_a, string_b)
- logical `op_ne_VS_VS` (string_a, string_b)
- logical `op_ne_CH_VS` (string_a, string_b)
- logical `op_ne_VS_CH` (string_a, string_b)
- logical `op_lt_VS_VS` (string_a, string_b)
- logical `op_lt_CH_VS` (string_a, string_b)
- logical `op_lt_VS_CH` (string_a, string_b)
- logical `op_le_VS_VS` (string_a, string_b)
- logical `op_le_CH_VS` (string_a, string_b)
- logical `op_le_VS_CH` (string_a, string_b)
- logical `op_ge_VS_VS` (string_a, string_b)
- logical `op_ge_CH_VS` (string_a, string_b)
- logical `op_ge_VS_CH` (string_a, string_b)
- logical `op_gt_VS_VS` (string_a, string_b)
- logical `op_gt_CH_VS` (string_a, string_b)
- logical `op_gt_VS_CH` (string_a, string_b)
- type(varying_string) `adjustl_` (string)

7.71.1 Detailed Description

Definition at line 65 of file iso_varying_string.f90.

7.71.2 Member Function Documentation

7.71.2.1 type(varying_string) ISO_VARYING_STRING::assignment(=)::adjustl_(type(varying_string),intent(in) string)

Definition at line 858 of file iso_varying_string.f90.

7.71.2.2 subroutine ISO_VARYING_STRING::assignment(=)::op_assign_CH_VS(character(LEN=*),intent(out) var, type(varying_string),intent(in) exp)

Definition at line 419 of file iso_varying_string.f90.

7.71.2.3 subroutine ISO_VARYING_STRING::assignment(=)::op_assign_VS_CH(type(varying_string),intent(out) var, character(LEN=*),intent(in) exp)

Definition at line 436 of file iso_varying_string.f90.

**7.71.2.4 type(varying_string) ISO_VARYING_STRING::assignment(=)::op_concat_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 484 of file iso_varying_string.f90.

**7.71.2.5 type(varying_string) ISO_VARYING_STRING::assignment(=)::op_concat_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 503 of file iso_varying_string.f90.

**7.71.2.6 type(varying_string) ISO_VARYING_STRING::assignment(=)::op_concat_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 453 of file iso_varying_string.f90.

**7.71.2.7 logical ISO_VARYING_STRING::assignment(=)::op_eq_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 540 of file iso_varying_string.f90.

**7.71.2.8 logical ISO_VARYING_STRING::assignment(=)::op_eq_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 559 of file iso_varying_string.f90.

**7.71.2.9 logical ISO_VARYING_STRING::assignment(=)::op_eq_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 522 of file iso_varying_string.f90.

**7.71.2.10 logical ISO_VARYING_STRING::assignment(=)::op_ge_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 764 of file iso_varying_string.f90.

**7.71.2.11 logical ISO_VARYING_STRING::assignment(=)::op_ge_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 783 of file iso_varying_string.f90.

**7.71.2.12 logical ISO_VARYING_STRING::assignment(=)::op_ge_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 746 of file iso_varying_string.f90.

**7.71.2.13 logical ISO_VARYING_STRING::assignment(=)::op_gt_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 820 of file iso_varying_string.f90.

**7.71.2.14 logical ISO_VARYING_STRING::assignment(=)::op_gt_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 839 of file iso_varying_string.f90.

**7.71.2.15 logical ISO_VARYING_STRING::assignment(=)::op_gt_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 802 of file iso_varying_string.f90.

**7.71.2.16 logical ISO_VARYING_STRING::assignment(=)::op_le_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 708 of file iso_varying_string.f90.

**7.71.2.17 logical ISO_VARYING_STRING::assignment(=)::op_le_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 727 of file iso_varying_string.f90.

**7.71.2.18 logical ISO_VARYING_STRING::assignment(=)::op_le_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 690 of file iso_varying_string.f90.

**7.71.2.19 logical ISO_VARYING_STRING::assignment(=)::op_lt_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 652 of file iso_varying_string.f90.

**7.71.2.20 logical ISO_VARYING_STRING::assignment(=)::op_lt_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 671 of file iso_varying_string.f90.

**7.71.2.21 logical ISO_VARYING_STRING::assignment(=)::op_lt_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 634 of file iso_varying_string.f90.

**7.71.2.22 logical ISO_VARYING_STRING::assignment(=)::op_ne_CH_VS
(character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 596 of file iso_varying_string.f90.

**7.71.2.23 logical ISO_VARYING_STRING::assignment(=)::op_ne_VS_CH
(type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)**

Definition at line 615 of file iso_varying_string.f90.

**7.71.2.24 logical ISO_VARYING_STRING::assignment(=)::op_ne_VS_VS
(type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)**

Definition at line 578 of file iso_varying_string.f90.

7.72 ISO_VARYING_STRING::char Interface Reference

Public Member Functions

- function `char_auto` (*string*)
- character(LEN=*length*) `char_fixed` (*string*, *length*)

7.72.1 Detailed Description

Definition at line 120 of file iso_varying_string.f90.

7.72.2 Member Function Documentation

7.72.2.1 `function ISO_VARYING_STRING::char::char_auto (type(varying_string),intent(in) string)`

Definition at line 892 of file iso_varying_string.f90.

7.72.2.2 `character(LEN=length) ISO_VARYING_STRING::char::char_fixed (type(varying_string),intent(in) string, integer,intent(in) length)`

Definition at line 914 of file iso_varying_string.f90.

7.73 ISO_VARYING_STRING::extract Interface Reference

Public Member Functions

- type(varying_string) [extract_VS](#) (string, start, finish)
- type(varying_string) [extract_CH](#) (string, start, finish)

7.73.1 Detailed Description

Definition at line 218 of file iso_varying_string.f90.

7.73.2 Member Function Documentation

7.73.2.1 type(varying_string) ISO_VARYING_STRING::extract::extract_CH (character(LEN=*)**,intent(in)** string, integer,intent(in),optional start, integer,intent(in),optional finish)

Definition at line 1901 of file iso_varying_string.f90.

7.73.2.2 type(varying_string) ISO_VARYING_STRING::extract::extract_VS (type(varying_string),**intent(in)** string, integer,intent(in),optional start, integer,intent(in),optional finish)

Definition at line 1882 of file iso_varying_string.f90.

7.74 ISO_VARYING_STRING::get Interface Reference

Public Member Functions

- subroutine `get_` (string, maxlen, iostat)
- subroutine `get_unit` (unit, string, maxlen, iostat)
- subroutine `get_set_VS` (string, set, separator, maxlen, iostat)
- subroutine `get_set_CH` (string, set, separator, maxlen, iostat)
- subroutine `get_unit_set_VS` (unit, string, set, separator, maxlen, iostat)
- subroutine `get_unit_set_CH` (unit, string, set, separator, maxlen, iostat)

7.74.1 Detailed Description

Definition at line 195 of file iso_varying_string.f90.

7.74.2 Member Function Documentation

7.74.2.1 subroutine ISO_VARYING_STRING::get::get_ (type(varying_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)

Definition at line 1455 of file iso_varying_string.f90.

7.74.2.2 subroutine ISO_VARYING_STRING::get::get_set_CH (type(varying_string),intent(out) *string*, character(LEN=*)intent(in) *set*, type(varying_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)

Definition at line 1583 of file iso_varying_string.f90.

7.74.2.3 subroutine ISO_VARYING_STRING::get::get_set_VS (type(varying_string),intent(out) *string*, type(varying_string),intent(in) *set*, type(varying_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)

Definition at line 1562 of file iso_varying_string.f90.

7.74.2.4 subroutine ISO_VARYING_STRING::get::get_unit (integer,intent(in) *unit*, type(varying_string),intent(out) *string*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)

Definition at line 1508 of file iso_varying_string.f90.

7.74.2.5 subroutine ISO_VARYING_STRING::get::get_unit_set_CH (integer,intent(in) *unit*, type(varying_string),intent(out) *string*, character(LEN=*)intent(in) *set*, type(varying_string),intent(out),optional *separator*, integer,intent(in),optional *maxlen*, integer,intent(out),optional *iostat*)

Definition at line 1663 of file iso_varying_string.f90.

**7.74.2.6 subroutine ISO_VARYING_STRING::get::get_unit_set_VS (integer,intent(in)
unit, type(varying_string),intent(out) string, type(varying_string),intent(in) set,
type(varying_string),intent(out),optional separator, integer,intent(in),optional maxlen,
integer,intent(out),optional iostat)**

Definition at line 1641 of file iso_varying_string.f90.

7.75 ISO_VARYING_STRING::iachar Interface Reference

Public Member Functions

- `integer iachar_(c)`

7.75.1 Detailed Description

Definition at line 125 of file iso_varying_string.f90.

7.75.2 Member Function Documentation

7.75.2.1 `integer ISO_VARYING_STRING::iachar::iachar_(type(varying_string),intent(in) c)`

Definition at line 933 of file iso_varying_string.f90.

7.76 ISO_VARYING_STRING::ichar Interface Reference

Public Member Functions

- `integer ichar_(c)`

7.76.1 Detailed Description

Definition at line 129 of file iso_varying_string.f90.

7.76.2 Member Function Documentation

7.76.2.1 `integer ISO_VARYING_STRING::ichar::ichar_(type(varying_string),intent(in) c)`

Definition at line 951 of file iso_varying_string.f90.

7.77 ISO_VARYING_STRING::index Interface Reference

Public Member Functions

- `integer index_VS_VS` (`string, substring, back`)
- `integer index_CH_VS` (`string, substring, back`)
- `integer index_VS_CH` (`string, substring, back`)

7.77.1 Detailed Description

Definition at line 133 of file iso_varying_string.f90.

7.77.2 Member Function Documentation

7.77.2.1 `integer ISO_VARYING_STRING::index::index_CH_VS` (`character(LEN=*)`,`intent(in)` *string*, `type(varying_string)`,`intent(in)` *substring*, `logical,intent(in),optional` *back*)

Definition at line 989 of file iso_varying_string.f90.

7.77.2.2 `integer ISO_VARYING_STRING::index::index_VS_CH` (`type(varying_string)`,`intent(in)` *string*, `character(LEN=*)`,`intent(in)` *substring*, `logical,intent(in),optional` *back*)

Definition at line 1009 of file iso_varying_string.f90.

7.77.2.3 `integer ISO_VARYING_STRING::index::index_VS_VS` (`type(varying_string)`,`intent(in)` *string*, `type(varying_string)`,`intent(in)` *substring*, `logical,intent(in),optional` *back*)

Definition at line 969 of file iso_varying_string.f90.

7.78 ISO_VARYING_STRING::insert Interface Reference

Public Member Functions

- type(varying_string) [insert_VS_VS](#) (string, start, substring)
- type(varying_string) [insert_CH_VS](#) (string, start, substring)
- type(varying_string) [insert_VS_CH](#) (string, start, substring)
- type(varying_string) [insert_CH_CH](#) (string, start, substring)

7.78.1 Detailed Description

Definition at line 223 of file iso_varying_string.f90.

7.78.2 Member Function Documentation

7.78.2.1 type(varying_string) ISO_VARYING_STRING::insert::insert_CH_CH (character(LEN=*),intent(in) string, integer,intent(in) start, character(LEN=*),intent(in) substring)

Definition at line 1992 of file iso_varying_string.f90.

7.78.2.2 type(varying_string) ISO_VARYING_STRING::insert::insert_CH_VS (character(LEN=*),intent(in) string, integer,intent(in) start, type(varying_string),intent(in) substring)

Definition at line 1954 of file iso_varying_string.f90.

7.78.2.3 type(varying_string) ISO_VARYING_STRING::insert::insert_VS_CH (type(varying_- string),intent(in) string, integer,intent(in) start, character(LEN=*),intent(in) substring)

Definition at line 1973 of file iso_varying_string.f90.

7.78.2.4 type(varying_string) ISO_VARYING_STRING::insert::insert_VS_VS (type(varying_- string),intent(in) string, integer,intent(in) start, type(varying_string),intent(in) substring)

Definition at line 1935 of file iso_varying_string.f90.

7.79 ISO_VARYING_STRING::len Interface Reference

Public Member Functions

- `integer len_(string)`

7.79.1 Detailed Description

Definition at line 139 of file iso_varying_string.f90.

7.79.2 Member Function Documentation

7.79.2.1 `integer ISO_VARYING_STRING::len::len_ (type(varying_string),intent(in) string)`

Definition at line 398 of file iso_varying_string.f90.

7.80 ISO_VARYING_STRING::len_trim Interface Reference

Public Member Functions

- `integer len_trim_(string)`

7.80.1 Detailed Description

Definition at line 143 of file iso_varying_string.f90.

7.80.2 Member Function Documentation

7.80.2.1 `integer ISO_VARYING_STRING::len_trim::len_trim_ (type(varying_string),intent(in) string)`

Definition at line 1029 of file iso_varying_string.f90.

7.81 ISO_VARYING_STRING::lge Interface Reference

Public Member Functions

- [logical lge_VS_VS \(string_a, string_b\)](#)
- [logical lge_CH_VS \(string_a, string_b\)](#)
- [logical lge_VS_CH \(string_a, string_b\)](#)

7.81.1 Detailed Description

Definition at line 147 of file iso_varying_string.f90.

7.81.2 Member Function Documentation

7.81.2.1 logical ISO_VARYING_STRING::lge::lge_CH_VS (character(LEN=*)**,intent(in)** *string_a*, type(varying_string)**,intent(in)** *string_b*)

Definition at line 1068 of file iso_varying_string.f90.

7.81.2.2 logical ISO_VARYING_STRING::lge::lge_VS_CH (type(varying_string)**,intent(in)** *string_a*, character(LEN=*)**,intent(in)** *string_b*)

Definition at line 1087 of file iso_varying_string.f90.

7.81.2.3 logical ISO_VARYING_STRING::lge::lge_VS_VS (type(varying_string)**,intent(in)** *string_a*, type(varying_string)**,intent(in)** *string_b*)

Definition at line 1050 of file iso_varying_string.f90.

7.82 ISO_VARYING_STRING::lgt Interface Reference

Public Member Functions

- [logical lgt_VS_VS](#) (string_a, string_b)
- [logical lgt_CH_VS](#) (string_a, string_b)
- [logical lgt_VS_CH](#) (string_a, string_b)

7.82.1 Detailed Description

Definition at line 153 of file iso_varying_string.f90.

7.82.2 Member Function Documentation

7.82.2.1 logical ISO_VARYING_STRING::lgt::lgt_CH_VS (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)

Definition at line 1124 of file iso_varying_string.f90.

7.82.2.2 logical ISO_VARYING_STRING::lgt::lgt_VS_CH (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)

Definition at line 1143 of file iso_varying_string.f90.

7.82.2.3 logical ISO_VARYING_STRING::lgt::lgt_VS_VS (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)

Definition at line 1106 of file iso_varying_string.f90.

7.83 ISO_VARYING_STRING::lle Interface Reference

Public Member Functions

- [logical lle_VS_VS](#) (string_a, string_b)
- [logical lle_CH_VS](#) (string_a, string_b)
- [logical lle_VS_CH](#) (string_a, string_b)

7.83.1 Detailed Description

Definition at line 159 of file iso_varying_string.f90.

7.83.2 Member Function Documentation

7.83.2.1 logical ISO_VARYING_STRING::lle::lle_CH_VS (character(LEN=*)**,intent(in)** *string_a*, type(varying_string)**,intent(in)** *string_b*)

Definition at line 1180 of file iso_varying_string.f90.

7.83.2.2 logical ISO_VARYING_STRING::lle::lle_VS_CH (type(varying_string)**,intent(in)** *string_a*, character(LEN=*)**,intent(in)** *string_b*)

Definition at line 1199 of file iso_varying_string.f90.

7.83.2.3 logical ISO_VARYING_STRING::lle::lle_VS_VS (type(varying_string)**,intent(in)** *string_a*, type(varying_string)**,intent(in)** *string_b*)

Definition at line 1162 of file iso_varying_string.f90.

7.84 ISO_VARYING_STRING::llt Interface Reference

Public Member Functions

- [logical llt_VS_VS](#) (string_a, string_b)
- [logical llt_CH_VS](#) (string_a, string_b)
- [logical llt_VS_CH](#) (string_a, string_b)

7.84.1 Detailed Description

Definition at line 165 of file iso_varying_string.f90.

7.84.2 Member Function Documentation

7.84.2.1 logical ISO_VARYING_STRING::llt::llt_CH_VS (character(LEN=*),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)

Definition at line 1236 of file iso_varying_string.f90.

7.84.2.2 logical ISO_VARYING_STRING::llt::llt_VS_CH (type(varying_string),intent(in) *string_a*, character(LEN=*),intent(in) *string_b*)

Definition at line 1255 of file iso_varying_string.f90.

7.84.2.3 logical ISO_VARYING_STRING::llt::llt_VS_VS (type(varying_string),intent(in) *string_a*, type(varying_string),intent(in) *string_b*)

Definition at line 1218 of file iso_varying_string.f90.

7.85 ISO_VARYING_STRING::put Interface Reference

Public Member Functions

- subroutine [put_VS](#) (string, iostat)
- subroutine [put_CH](#) (string, iostat)
- subroutine [put_unit_VS](#) (unit, string, iostat)
- subroutine [put_unit_CH](#) (unit, string, iostat)

7.85.1 Detailed Description

Definition at line 204 of file iso_varying_string.f90.

7.85.2 Member Function Documentation

7.85.2.1 subroutine ISO_VARYING_STRING::put::put_CH (character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1738 of file iso_varying_string.f90.

7.85.2.2 subroutine ISO_VARYING_STRING::put::put_unit_CH (integer,intent(in) *unit*, character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1777 of file iso_varying_string.f90.

7.85.2.3 subroutine ISO_VARYING_STRING::put::put_unit_VS (integer,intent(in) *unit*, type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1758 of file iso_varying_string.f90.

7.85.2.4 subroutine ISO_VARYING_STRING::put::put_VS (type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1722 of file iso_varying_string.f90.

7.86 ISO_VARYING_STRING::put_line Interface Reference

Public Member Functions

- subroutine [put_line_VS](#) (string, iostat)
- subroutine [put_line_CH](#) (string, iostat)
- subroutine [put_line_unit_VS](#) (unit, string, iostat)
- subroutine [put_line_unit_CH](#) (unit, string, iostat)

7.86.1 Detailed Description

Definition at line 211 of file iso_varying_string.f90.

7.86.2 Member Function Documentation

7.86.2.1 subroutine ISO_VARYING_STRING::put_line::put_line_CH (character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1818 of file iso_varying_string.f90.

7.86.2.2 subroutine ISO_VARYING_STRING::put_line::put_line_unit_CH (integer,intent(in) *unit*, character(LEN=*),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1859 of file iso_varying_string.f90.

7.86.2.3 subroutine ISO_VARYING_STRING::put_line::put_line_unit_VS (integer,intent(in) *unit*, type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1840 of file iso_varying_string.f90.

7.86.2.4 subroutine ISO_VARYING_STRING::put_line::put_line_VS (type(varying_string),intent(in) *string*, integer,intent(out),optional *iostat*)

Definition at line 1800 of file iso_varying_string.f90.

7.87 ISO_VARYING_STRING::remove Interface Reference

Public Member Functions

- TYPE(varying_string) **remove_VS** (string, start, finish)
- type(varying_string) **remove_CH** (string, start, finish)

7.87.1 Detailed Description

Definition at line 230 of file iso_varying_string.f90.

7.87.2 Member Function Documentation

7.87.2.1 **type(varying_string) ISO_VARYING_STRING::remove::remove_CH** (character(LEN=*)**,intent(in)** *string*, integer,**intent(in)**,optional *start*, integer,**intent(in)**,optional *finish*)

Definition at line 2035 of file iso_varying_string.f90.

7.87.2.2 **TYPE(varying_string) ISO_VARYING_STRING::remove::remove_VS** (type(varying_string),**intent(in)** *string*, integer,**intent(in)**,optional *start*, integer,**intent(in)**,optional *finish*)

Definition at line 2016 of file iso_varying_string.f90.

7.88 ISO_VARYING_STRING::repeat Interface Reference

Public Member Functions

- type(varying_string) `repeat_` (string, ncopies)

7.88.1 Detailed Description

Definition at line 171 of file iso_varying_string.f90.

7.88.2 Member Function Documentation

7.88.2.1 type(varying_string) ISO_VARYING_STRING::repeat::repeat_(type(varying_string),intent(in) string, integer,intent(in) ncopies)

Definition at line 1274 of file iso_varying_string.f90.

7.89 ISO_VARYING_STRING::replace Interface Reference

Public Member Functions

- type(varying_string) `replace_VS_VS_auto` (string, start, substring)
- type(varying_string) `replace_CH_VS_auto` (string, start, substring)
- type(varying_string) `replace_VS_CH_auto` (string, start, substring)
- type(varying_string) `replace_CH_CH_auto` (string, start, substring)
- type(varying_string) `replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying_string) `replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying_string) `replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying_string) `replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying_string) `replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying_string) `replace_CH_CH_CH_target` (string, target, substring, every, back)

7.89.1 Detailed Description

Definition at line 235 of file iso_varying_string.f90.

7.89.2 Member Function Documentation

7.89.2.1 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_CH_auto
`(character(LEN=*)intent(in) string, integer,intent(in) start, character(LEN=*)intent(in) substring)`

Definition at line 2133 of file iso_varying_string.f90.

7.89.2.2 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_CH_CH_target
`(character(LEN=*)intent(in) string, character(LEN=*)intent(in) target, character(LEN=*)intent(in) substring, logical,intent(in),optional every, logical,intent(in),optional back)`

Definition at line 2410 of file iso_varying_string.f90.

7.89.2.3 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_CH_fixed
`(character(LEN=*)intent(in) string, integer,intent(in) start, integer,intent(in) finish, character(LEN=*)intent(in) substring)`

Definition at line 2218 of file iso_varying_string.f90.

7.89.2.4 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_CH_VS_target
 (character(LEN=*),intent(in) *string*, character(LEN=*),intent(in) *target*,
 type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*)

Definition at line 2318 of file iso_varying_string.f90.

7.89.2.5 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_-
VS_auto (character(LEN=*),intent(in) *string*, integer,intent(in) *start*,
 type(varying_string),intent(in) *substring*)

Definition at line 2093 of file iso_varying_string.f90.

7.89.2.6 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_VS_CH_target
 (character(LEN=*),intent(in) *string*, type(varying_string),intent(in) *target*,
 character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*)

Definition at line 2364 of file iso_varying_string.f90.

7.89.2.7 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_VS_fixed
 (character(LEN=*),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
 type(varying_string),intent(in) *substring*)

Definition at line 2176 of file iso_varying_string.f90.

7.89.2.8 type(varying_string) ISO_VARYING_STRING::replace::replace_CH_VS_VS_target
 (character(LEN=*),intent(in) *string*, type(varying_string),intent(in) *target*,
 type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*)

Definition at line 2272 of file iso_varying_string.f90.

7.89.2.9 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_-
CH_auto (type(varying_string),intent(in) *string*, integer,intent(in) *start*,
 character(LEN=*),intent(in) *substring*)

Definition at line 2113 of file iso_varying_string.f90.

7.89.2.10 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_CH_CH_target
 (type(varying_string),intent(in) *string*, character(LEN=*),intent(in) *target*,
 character(LEN=*),intent(in) *substring*, logical,intent(in),optional *every*,
 logical,intent(in),optional *back*)

Definition at line 2387 of file iso_varying_string.f90.

**7.89.2.11 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_CH_fixed
(type(varying_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
character(LEN=*)intent(in) *substring*)**

Definition at line 2197 of file iso_varying_string.f90.

**7.89.2.12 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_CH_VS_target
(type(varying_string),intent(in) *string*, character(LEN=*)intent(in) *target*,
type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*)**

Definition at line 2295 of file iso_varying_string.f90.

**7.89.2.13 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_VS_auto
(type(varying_string),intent(in) *string*, integer,intent(in) *start*,
type(varying_string),intent(in) *substring*)**

Definition at line 2073 of file iso_varying_string.f90.

**7.89.2.14 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_VS_CH_target
(type(varying_string),intent(in) *string*, type(varying_string),intent(in) *target*,
character(LEN=*)intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*)**

Definition at line 2341 of file iso_varying_string.f90.

**7.89.2.15 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_VS_fixed
(type(varying_string),intent(in) *string*, integer,intent(in) *start*, integer,intent(in) *finish*,
type(varying_string),intent(in) *substring*)**

Definition at line 2153 of file iso_varying_string.f90.

**7.89.2.16 type(varying_string) ISO_VARYING_STRING::replace::replace_VS_VS_VS_target
(type(varying_string),intent(in) *string*, type(varying_string),intent(in) *target*,
type(varying_string),intent(in) *substring*, logical,intent(in),optional *every*,
logical,intent(in),optional *back*)**

Definition at line 2249 of file iso_varying_string.f90.

7.90 ISO_VARYING_STRING::scan Interface Reference

Public Member Functions

- `integer scan_VS_VS (string, set, back)`
- `integer scan_CH_VS (string, set, back)`
- `integer scan_VS_CH (string, set, back)`

7.90.1 Detailed Description

Definition at line 175 of file iso_varying_string.f90.

7.90.2 Member Function Documentation

7.90.2.1 `integer ISO_VARYING_STRING::scan::scan_CH_VS (character(LEN=*)&,intent(in) string, type(varying_string)&,intent(in) set, logical,intent(in),optional back)`

Definition at line 1312 of file iso_varying_string.f90.

7.90.2.2 `integer ISO_VARYING_STRING::scan::scan_VS_CH (type(varying_string)&,intent(in) string, character(LEN=*)&,intent(in) set, logical,intent(in),optional back)`

Definition at line 1332 of file iso_varying_string.f90.

7.90.2.3 `integer ISO_VARYING_STRING::scan::scan_VS_VS (type(varying_string)&,intent(in) string, type(varying_string)&,intent(in) set, logical,intent(in),optional back)`

Definition at line 1292 of file iso_varying_string.f90.

7.91 ISO_VARYING_STRING::split Interface Reference

Public Member Functions

- subroutine [split_VS](#) (string, word, set, separator, back)
- subroutine [split_CH](#) (string, word, set, separator, back)

7.91.1 Detailed Description

Definition at line 254 of file iso_varying_string.f90.

7.91.2 Member Function Documentation

7.91.2.1 subroutine ISO_VARYING_STRING::split::split_CH (*type(varying_string),intent(inout) string*, *type(varying_string),intent(out) word*, *character(LEN=*)intent(in) set*,
type(varying_string),intent(out),optional separator, *logical,intent(in),optional back*)

Definition at line 2514 of file iso_varying_string.f90.

7.91.2.2 subroutine ISO_VARYING_STRING::split::split_VS (*type(varying_string),intent(inout) string*, *type(varying_string),intent(out) word*, *type(varying_string),intent(in) set*,
type(varying_string),intent(out),optional separator, *logical,intent(in),optional back*)

Definition at line 2494 of file iso_varying_string.f90.

7.92 ISO_VARYING_STRING::trim Interface Reference

Public Member Functions

- type(varying_string) **trim_** (string)

7.92.1 Detailed Description

Definition at line 181 of file iso_varying_string.f90.

7.92.2 Member Function Documentation

7.92.2.1 type(varying_string) ISO_VARYING_STRING::trim::trim_ (type(varying_string),intent(in) string)

Definition at line 1352 of file iso_varying_string.f90.

7.93 ISO_VARYING_STRING::var_str Interface Reference

Public Member Functions

- type(varying_string) [var_str_ \(char\)](#)

7.93.1 Detailed Description

Definition at line 191 of file iso_varying_string.f90.

7.93.2 Member Function Documentation

7.93.2.1 type(varying_string) ISO_VARYING_STRING::var_str::var_str_(character(LEN=*),intent(in) *char*)

Definition at line 1429 of file iso_varying_string.f90.

7.94 ISO_VARYING_STRING::VARYING_STRING Struct Reference

Public Attributes

- character(LEN=1), dimension(:), allocatable [chars](#)

7.94.1 Detailed Description

Definition at line 58 of file iso_varying_string.f90.

7.94.2 Member Data Documentation

7.94.2.1 character(LEN=1),dimension(:),allocatable ISO_VARYING_STRING::VARYING_STRING::chars

Definition at line 60 of file iso_varying_string.f90.

7.95 ISO_VARYING_STRING::verify Interface Reference

Public Member Functions

- [integer verify_VS_VS \(string, set, back\)](#)
- [integer verify_CH_VS \(string, set, back\)](#)
- [integer verify_VS_CH \(string, set, back\)](#)

7.95.1 Detailed Description

Definition at line 185 of file iso_varying_string.f90.

7.95.2 Member Function Documentation

7.95.2.1 [integer ISO_VARYING_STRING::verify::verify_CH_VS \(character\(LEN=*\)&,intent\(in\) string, type\(varying_string\)&,intent\(in\) set, logical,intent\(in\),optional back\)](#)

Definition at line 1389 of file iso_varying_string.f90.

7.95.2.2 [integer ISO_VARYING_STRING::verify::verify_VS_CH \(type\(varying_string\)&,intent\(in\) string, character\(LEN=*\)&,intent\(in\) set, logical,intent\(in\),optional back\)](#)

Definition at line 1409 of file iso_varying_string.f90.

7.95.2.3 [integer ISO_VARYING_STRING::verify::verify_VS_VS \(type\(varying_string\)&,intent\(in\) string, type\(varying_string\)&,intent\(in\) set, logical,intent\(in\),optional back\)](#)

Definition at line 1369 of file iso_varying_string.f90.

7.96 LAPACK::interface Interface Reference

Public Member Functions

- subroutine [DGESV](#) (N, NRHS, A, LDA, IPIV, B, LDB, INFO)

7.96.1 Detailed Description

Definition at line 50 of file lapack.f90.

7.96.2 Member Function Documentation

- 7.96.2.1 subroutine **LAPACK::interface::DGESV** (**INTEGER(INTG),intent(in) N,**
INTEGER(INTG),intent(in) NRHS, **REAL(DP),dimension(lda,*),intent(inout) A,**
INTEGER(INTG),intent(in) LDA, **INTEGER(INTG),dimension(*),intent(out) IPIV,**
REAL(DP),dimension(lbd,*),intent(inout) B, **INTEGER(INTG),intent(in) LDB,**
INTEGER(INTG),intent(out) INFO)

Definition at line 52 of file lapack.f90.

7.97 LISTS::LIST_DETACH_AND_DESTROY Interface Reference

Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.

Private Member Functions

- subroutine [LIST_DETACH_AND_DESTROY_INTG](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)
- subroutine [LIST_DETACH_AND_DESTROY_SP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)
- subroutine [LIST_DETACH_AND_DESTROY_DP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

7.97.1 Detailed Description

Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.

See also:

[LISTS](#).

Definition at line 126 of file lists.f90.

7.97.2 Member Function Documentation

7.97.2.1 subroutine LISTS::LIST_DETACH_AND_DESTROY::LIST_DETACH_AND_DESTROY_DP (TYPE(LIST_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *NUMBER_IN_LIST*, REAL(DP),dimension(:),pointer *LIST_VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)
[private]

Parameters:

LIST The pointer to the list

NUMBER_IN_LIST On exit, the number in the list that has been detached.

LIST_VALUES On exit, a pointer to the detached list. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 931 of file lists.f90.

7.97.2.2 subroutine LISTS::LIST_DETACH_AND_DESTROY::LIST_DETACH_AND_DESTROY_INTG (TYPE(LIST_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *NUMBER_IN_LIST*, INTEGER(INTG),dimension(:),pointer *LIST_VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

LIST The pointer to the list

NUMBER_IN_LIST On exit, the number in the list that has been detached.

LIST_VALUES On exit, a pointer to the detached list. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 830 of file lists.f90.

7.97.2.3 subroutine LISTS::LIST_DETACH_AND_DESTROY::LIST_DETACH_AND_DESTROY_SP (TYPE(LIST_TYPE),pointer *LIST*, INTEGER(INTG),intent(out) *NUMBER_IN_LIST*, REAL(SP),dimension(:),pointer *LIST_VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

LIST The pointer to the list

NUMBER_IN_LIST On exit, the number in the list that has been detached.

LIST_VALUES On exit, a pointer to the detached list. Must not be associated on entry.

ERR The error code

ERROR The error string

Definition at line 881 of file lists.f90.

7.98 LISTS::LIST_ITEM_ADD Interface Reference

Adds an item to the end of a list.

Private Member Functions

- subroutine [LIST_ITEM_ADD_INTG1](#) (LIST, ITEM, ERR, ERROR,*)
- subroutine [LIST_ITEM_ADD_SP1](#) (LIST, ITEM, ERR, ERROR,*)
- subroutine [LIST_ITEM_ADD_DP1](#) (LIST, ITEM, ERR, ERROR,*)

7.98.1 Detailed Description

Adds an item to the end of a list.

See also:

[LISTS](#).

Definition at line 112 of file lists.f90.

7.98.2 Member Function Documentation

7.98.2.1 subroutine LISTS::LIST_ITEM_ADD::LIST_ITEM_ADD_DP1 (TYPE(LIST_TYPE),pointer LIST, REAL(DP),intent(in) ITEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

LIST A pointer to the list

ITEM The item to add

ERR The error code

ERROR The error string

Definition at line 566 of file lists.f90.

7.98.2.2 subroutine LISTS::LIST_ITEM_ADD::LIST_ITEM_ADD_INTG1 (TYPE(LIST_TYPE),pointer LIST, INTEGER(INTG),intent(in) ITEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

LIST A pointer to the list

ITEM The item to add

ERR The error code

ERROR The error string

Definition at line 457 of file lists.f90.

7.98.2.3 subroutine LISTS::LIST_ITEM_ADD::LIST_ITEM_ADD_SP1 (TYPE(LIST_-
TYPE),pointer *LIST*, REAL(SP),intent(in) *ITEM*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

LIST A pointer to the list

ITEM The item to add

ERR The error code

ERROR The error string

Definition at line 511 of file lists.f90.

7.99 LISTS::LIST_ITEM_IN_LIST Interface Reference

Determines if an item is in a list and returns the position of the item.

Private Member Functions

- subroutine [LIST_ITEM_IN_LIST_INTG1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)
- subroutine [LIST_ITEM_IN_LIST_SP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)
- subroutine [LIST_ITEM_IN_LIST_DP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

7.99.1 Detailed Description

Determines if an item is in a list and returns the position of the item.

See also:

[LISTS](#).

Definition at line 119 of file lists.f90.

7.99.2 Member Function Documentation

7.99.2.1 subroutine LISTS::LIST_ITEM_IN_LIST::LIST_ITEM_IN_LIST_DP1

(**TYPE(LIST_TYPE),pointer LIST,** REAL(DP),intent(in) **ITEM,**
INTEGER(INTG),intent(out) LIST_ITEM, INTEGER(INTG),intent(out) **ERR,**
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string

Definition at line 707 of file lists.f90.

7.99.2.2 subroutine LISTS::LIST_ITEM_IN_LIST::LIST_ITEM_IN_LIST_INTG1

(**TYPE(LIST_TYPE),pointer LIST,** INTEGER(INTG),intent(in) **ITEM,**
INTEGER(INTG),intent(out) LIST_ITEM, INTEGER(INTG),intent(out) **ERR,**
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string.

Definition at line 621 of file lists.f90.

7.99.2.3 subroutine LISTS::LIST_ITEM_IN_LIST::LIST_ITEM_IN_LIST_SP1
(**TYPE(LIST_TYPE),pointer LIST,** **REAL(SP),intent(in) ITEM,**
INTEGER(INTG),intent(out) LIST_ITEM, **INTEGER(INTG),intent(out) ERR,**
TYPE(VARYING_STRING),intent(out) ERROR, ***) [private]**

Parameters:

LIST The pointer to the list

ITEM The item to find.

LIST_ITEM On exit, the position of the item in the list. If the item does not exist then the value of 0 is returned.

ERR The error code

ERROR The error string

Definition at line 664 of file lists.f90.

7.100 LISTS::LIST_PTR_TYPE Struct Reference

Buffer type to allow arrays of pointers to a list.

Collaboration diagram for LISTS::LIST_PTR_TYPE:

Private Attributes

- TYPE([LIST_TYPE](#)), pointer PTR

The pointer to the list.

7.100.1 Detailed Description

Buffer type to allow arrays of pointers to a list.

Definition at line 89 of file lists.f90.

7.100.2 Member Data Documentation

7.100.2.1 TYPE(LIST_TYPE),pointer LISTS::LIST_PTR_TYPE::PTR [private]

The pointer to the list.

Definition at line 90 of file lists.f90.

7.101 LISTS::LIST_SEARCH Interface Reference

Searches a list for a given value and returns the position in the list if the value exists.

Private Member Functions

- subroutine [LIST_SEARCH_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)
- subroutine [LIST_SEARCH_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)
- subroutine [LIST_SEARCH_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

7.101.1 Detailed Description

Searches a list for a given value and returns the position in the list if the value exists.

See also:

[LISTS](#).

Definition at line 133 of file lists.f90.

7.101.2 Member Function Documentation

7.101.2.1 subroutine LISTS::LIST_SEARCH::LIST_SEARCH_DP_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1157 of file lists.f90.

7.101.2.2 subroutine LISTS::LIST_SEARCH::LIST_SEARCH_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1103 of file lists.f90.

7.101.2.3 subroutine LISTS::LIST_SEARCH::LIST_SEARCH_SP_ARRAY
(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),intent(in) *VALUE*,
INTEGER(INTG),intent(out) *POSITION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1130 of file lists.f90.

7.102 LISTS::LIST_SEARCH_LINEAR Interface Reference

Searches a list using the linear search method.

Private Member Functions

- subroutine [LIST_SEARCH_LINEAR_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)
- subroutine [LIST_SEARCH_LINEAR_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)
- subroutine [LIST_SEARCH_LINEAR_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

7.102.1 Detailed Description

Searches a list using the linear search method.

Definition at line 140 of file lists.f90.

7.102.2 Member Function Documentation

7.102.2.1 subroutine LISTS::LIST_SEARCH_LINEAR::LIST_SEARCH_LINEAR_DP_ARRAY (REAL(DP),dimension(:),intent(in) A, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1266 of file lists.f90.

7.102.2.2 subroutine LISTS::LIST_SEARCH_LINEAR::LIST_SEARCH_LINEAR_INTG_- ARRAY (INTEGER(INTG),dimension(:),intent(in) A, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) POSITION, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1184 of file lists.f90.

7.102.2.3 subroutine LISTS::LIST_SEARCH_LINEAR::LIST_SEARCH_LINEAR_SP_ARRAY
(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),intent(in) *VALUE*,
INTEGER(INTG),intent(out) *POSITION*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

A The list to search

VALUE The value to search for

POSITION On exit, the position of value in the list. If value does not exist in the list the returned position is zero.

ERR The error code

ERROR The error string

Definition at line 1225 of file lists.f90.

7.103 LISTS::LIST_SORT Interface Reference

Sorts a list into ascending order.

Private Member Functions

- subroutine `LIST_SORT_INTG_ARRAY` (*A, ERR, ERROR,**)
- subroutine `LIST_SORT_SP_ARRAY` (*A, ERR, ERROR,**)
- subroutine `LIST_SORT_DP_ARRAY` (*A, ERR, ERROR,**)

7.103.1 Detailed Description

Sorts a list into ascending order.

Definition at line 147 of file lists.f90.

7.103.2 Member Function Documentation

7.103.2.1 subroutine LISTS::LIST_SORT::LIST_SORT_DP_ARRAY `(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1357 of file lists.f90.

7.103.2.2 subroutine LISTS::LIST_SORT::LIST_SORT_INTG_ARRAY `(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Parameters:

- A*** The list to sort
- ERR*** The error code
- ERROR*** The error string

Definition at line 1307 of file lists.f90.

7.103.2.3 subroutine LISTS::LIST_SORT::LIST_SORT_SP_ARRAY `(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Parameters:

- A*** The list to sort
- ERR*** The error code

ERROR The error string

Definition at line 1332 of file lists.f90.

7.104 LISTS::LIST_SORT_BUBBLE Interface Reference

Sorts a list into assending order using the bubble sort method.

Private Member Functions

- subroutine [LIST_SORT_BUBBLE_INTG_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_BUBBLE_SP_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_BUBBLE_DP_ARRAY](#) (A, ERR, ERROR,*)

7.104.1 Detailed Description

Sorts a list into assending order using the bubble sort method.

Definition at line 154 of file lists.f90.

7.104.2 Member Function Documentation

7.104.2.1 subroutine LISTS::LIST_SORT_BUBBLE::LIST_SORT_BUBBLE_DP_ARRAY $(\text{REAL}(\text{DP}), \text{dimension}(:), \text{intent(inout)} A, \text{ INTEGER}(\text{INTG}), \text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING_STRING}), \text{intent(out)} \text{ERROR}, *)$ [private]

Parameters:

- A** The list to sort
ERR The error code
ERROR The error string

Definition at line 1463 of file lists.f90.

7.104.2.2 subroutine LISTS::LIST_SORT_BUBBLE::LIST_SORT_BUBBLE_INTG_ARRAY $(\text{INTEGER}(\text{INTG}), \text{dimension}(:), \text{intent(inout)} A, \text{ INTEGER}(\text{INTG}), \text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING_STRING}), \text{intent(out)} \text{ERROR}, *)$ [private]

Parameters:

- A** The list to sort
ERR The error code
ERROR The error string

Definition at line 1382 of file lists.f90.

7.104.2.3 subroutine LISTS::LIST_SORT_BUBBLE::LIST_SORT_BUBBLE_SP_ARRAY $(\text{REAL}(\text{SP}), \text{dimension}(:), \text{intent(inout)} A, \text{ INTEGER}(\text{INTG}), \text{intent(out)} \text{ERR},$ $\text{TYPE}(\text{VARYING_STRING}), \text{intent(out)} \text{ERROR}, *)$ [private]

Parameters:

- A** The list to sort
ERR The error code

ERROR The error string

Definition at line 1422 of file lists.f90.

7.105 LISTS::LIST_SORT_HEAP Interface Reference

Sorts a list into assending order using the heap sort method.

Private Member Functions

- subroutine [LIST_SORT_HEAP_INTG_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_HEAP_SP_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_HEAP_DP_ARRAY](#) (A, ERR, ERROR,*)

7.105.1 Detailed Description

Sorts a list into assending order using the heap sort method.

Definition at line 161 of file lists.f90.

7.105.2 Member Function Documentation

7.105.2.1 subroutine LISTS::LIST_SORT_HEAP::LIST_SORT_HEAP_DP_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1619 of file lists.f90.

7.105.2.2 subroutine LISTS::LIST_SORT_HEAP::LIST_SORT_HEAP_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

- A** The list to sort
- ERR** The error code
- ERROR** The error string

Definition at line 1504 of file lists.f90.

7.105.2.3 subroutine LISTS::LIST_SORT_HEAP::LIST_SORT_HEAP_SP_ARRAY (REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

- A** The list to sort
- ERR** The error code

ERROR The error string

Definition at line 1561 of file lists.f90.

7.106 LISTS::LIST_SORT_SHELL Interface Reference

Sorts a list into either assending or descending order using the shell sort method.

Private Member Functions

- subroutine [LIST_SORT_SHELL_INTG_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_SHELL_SP_ARRAY](#) (A, ERR, ERROR,*)
- subroutine [LIST_SORT_SHELL_DP_ARRAY](#) (A, ERR, ERROR,*)

7.106.1 Detailed Description

Sorts a list into either assending or descending order using the shell sort method.

Definition at line 168 of file lists.f90.

7.106.2 Member Function Documentation

7.106.2.1 subroutine LISTS::LIST_SORT_SHELL::LIST_SORT_SHELL_DP_ARRAY (REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

ERR The error code

ERROR The error string

Definition at line 1760 of file lists.f90.

7.106.2.2 subroutine LISTS::LIST_SORT_SHELL::LIST_SORT_SHELL_INTG_ARRAY (INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to sort

ERR The error code

ERROR The error string

Definition at line 1677 of file lists.f90.

7.106.2.3 subroutine LISTS::LIST_SORT_SHELL::LIST_SORT_SHELL_SP_ARRAY (REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

A The list to sort

ERR The error code

ERROR The error string

Definition at line 1718 of file lists.f90.

7.107 LISTS::LIST_TYPE Struct Reference

Contains information on a list.

Private Attributes

- LOGICAL [LIST_FINISHED](#)
Is .TRUE. if the list has finished being created, .FALSE. if not.
- INTEGER(INTG) [NUMBER_IN_LIST](#)
The number of items currently in the list.
- INTEGER(INTG) [INITIAL_SIZE](#)
The size of the list when it was initially created.
- INTEGER(INTG) [SIZE](#)
The current size of the list.
- INTEGER(INTG) [DATA_TYPE](#)
The data type of the list.
- INTEGER(INTG) [SORT_ORDER](#)
The ordering to be used when sorting the list.
- INTEGER(INTG) [SORT_METHOD](#)
The sorting method to be used when sorting the list.
- INTEGER(INTG), pointer [LIST_INTG](#)
A pointer to the integer data for integer lists.
- REAL(SP), pointer [LIST_SP](#)
A pointer to the single precision data for single precision real lists.
- REAL(DP), pointer [LIST_DP](#)
A pointer to the double precision data for double precision real lists.

7.107.1 Detailed Description

Contains information on a list.

Definition at line 94 of file lists.f90.

7.107.2 Member Data Documentation

7.107.2.1 INTEGER(INTG) LISTS::LIST_TYPE::DATA_TYPE [private]

The data type of the list.

See also:

[LISTS::DataType](#)

Definition at line 99 of file lists.f90.

7.107.2.2 INTEGER(INTG) LISTS::LIST_TYPE::INITIAL_SIZE [private]

The size of the list when it was initially created.

Definition at line 97 of file lists.f90.

7.107.2.3 REAL(DP),pointer LISTS::LIST_TYPE::LIST_DP [private]

A pointer to the double precision data for double precision real lists.

Definition at line 104 of file lists.f90.

7.107.2.4 LOGICAL LISTS::LIST_TYPE::LIST_FINISHED [private]

Is .TRUE. if the list has finished being created, .FALSE. if not.

Definition at line 95 of file lists.f90.

7.107.2.5 INTEGER(INTG),pointer LISTS::LIST_TYPE::LIST_INTG [private]

A pointer to the integer data for integer lists.

Definition at line 102 of file lists.f90.

7.107.2.6 REAL(SP),pointer LISTS::LIST_TYPE::LIST_SP [private]

A pointer to the single precision data for single precision real lists.

Definition at line 103 of file lists.f90.

7.107.2.7 INTEGER(INTG) LISTS::LIST_TYPE::NUMBER_IN_LIST [private]

The number of items currently in the list.

Definition at line 96 of file lists.f90.

7.107.2.8 INTEGER(INTG) LISTS::LIST_TYPE::SIZE [private]

The current size of the list.

Definition at line 98 of file lists.f90.

7.107.2.9 INTEGER(INTG) LISTS::LIST_TYPE::SORT_METHOD [private]

The sorting method to be used when sorting the list.

See also:

[LISTS_SortingMethod](#)

Definition at line 101 of file lists.f90.

7.107.2.10 INTEGER(INTG) LISTS::LIST_TYPE::SORT_ORDER [private]

The ordering to be used when sorting the list.

See also:

[LISTS::SortingOrder](#)

Definition at line 100 of file lists.f90.

7.108 MATHS::CROSS_PRODUCT Interface Reference

Private Member Functions

- subroutine [CROSS_PRODUCT_INTG](#) (A, B, C, ERR, ERROR,*)
- subroutine [CROSS_PRODUCT_SP](#) (A, B, C, ERR, ERROR,*)
- subroutine [CROSS_PRODUCT_DP](#) (A, B, C, ERR, ERROR,*)

7.108.1 Detailed Description

Definition at line 61 of file maths.f90.

7.108.2 Member Function Documentation

7.108.2.1 subroutine MATHS::CROSS_PRODUCT::CROSS_PRODUCT_DP
(REAL(DP),dimension(:),intent(in) *A*, REAL(DP),dimension(:),intent(in) *B*,
REAL(DP),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 239 of file maths.f90.

7.108.2.2 subroutine MATHS::CROSS_PRODUCT::CROSS_PRODUCT_INTG
(INTEGER(INTG),dimension(:),intent(in) *A*, INTEGER(INTG),dimension(:),intent(in)
B, INTEGER(INTG),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 151 of file maths.f90.

7.108.2.3 subroutine MATHS::CROSS_PRODUCT::CROSS_PRODUCT_SP
(REAL(SP),dimension(:),intent(in) *A*, REAL(SP),dimension(:),intent(in) *B*,
REAL(SP),dimension(:),intent(out) *C*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 195 of file maths.f90.

7.109 MATHS::D_CROSS_PRODUCT Interface Reference

Private Member Functions

- subroutine [D_CROSS_PRODUCT_INTG](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [D_CROSS_PRODUCT_SP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [D_CROSS_PRODUCT_DP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)

7.109.1 Detailed Description

Definition at line 67 of file maths.f90.

7.109.2 Member Function Documentation

7.109.2.1 subroutine MATHS::D_CROSS_PRODUCT::D_CROSS_PRODUCT_DP
`(INTEGER(INTG),intent(in) N, REAL(DP),dimension(:,),intent(in) A,
 REAL(DP),dimension(:,),intent(in) B, REAL(DP),dimension(:,),intent(out) C,
 REAL(DP),dimension(:, :,),intent(in) D_A, REAL(DP),dimension(:, :,),intent(in) D_B,
 REAL(DP),dimension(:, :,),intent(out) D_C, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 404 of file maths.f90.

7.109.2.2 subroutine MATHS::D_CROSS_PRODUCT::D_CROSS_PRODUCT_INTG
`(INTEGER(INTG),intent(in) N, INTEGER(INTG),dimension(:,),intent(in) A,
 INTEGER(INTG),dimension(:,),intent(in) B, INTEGER(INTG),dimension(:,),intent(out) C,
 INTEGER(INTG),dimension(:, :,),intent(in) D_A, INTEGER(INTG),dimension(:, :,),intent(in) D_B,
 INTEGER(INTG),dimension(:, :,),intent(out) D_C, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
 ERROR, *) [private]`

Definition at line 293 of file maths.f90.

7.109.2.3 subroutine MATHS::D_CROSS_PRODUCT::D_CROSS_PRODUCT_SP
`(INTEGER(INTG),intent(in) N, REAL(SP),dimension(:,),intent(in) A,
 REAL(SP),dimension(:,),intent(in) B, REAL(SP),dimension(:,),intent(out) C,
 REAL(SP),dimension(:, :,),intent(in) D_A, REAL(SP),dimension(:, :,),intent(in) D_B,
 REAL(SP),dimension(:, :,),intent(out) D_C, INTEGER(INTG),intent(out) ERR,
 TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Definition at line 348 of file maths.f90.

7.110 MATHS::DETERMINANT Interface Reference

Private Member Functions

- INTEGER(INTG) [DETERMINANT_FULL_INTG](#) (A, ERR, ERROR)
- REAL(SP) [DETERMINANT_FULL_SP](#) (A, ERR, ERROR)
- REAL(DP) [DETERMINANT_FULL_DP](#) (A, ERR, ERROR)

7.110.1 Detailed Description

Definition at line 73 of file maths.f90.

7.110.2 Member Function Documentation

7.110.2.1 REAL(DP) MATHS::DETERMINANT::DETERMINANT_FULL_DP
(REAL(DP),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 561 of file maths.f90.

7.110.2.2 INTEGER(INTG) MATHS::DETERMINANT::DETERMINANT_FULL_INTG
(INTEGER(INTG),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 469 of file maths.f90.

7.110.2.3 REAL(SP) MATHS::DETERMINANT::DETERMINANT_FULL_SP
(REAL(SP),dimension(:,:),intent(in) *A*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Definition at line 515 of file maths.f90.

7.111 MATHS::EDP Interface Reference

Private Member Functions

- REAL(DP) [EDP_DP](#) (X)
- REAL(SP) [EDP_SP](#) (X)

7.111.1 Detailed Description

Definition at line 79 of file maths.f90.

7.111.2 Member Function Documentation

7.111.2.1 REAL(DP) MATHS::EDP::EDP_DP (REAL(DP),intent(in) X) [private]

Definition at line 616 of file maths.f90.

7.111.2.2 REAL(SP) MATHS::EDP::EDP_SP (REAL(SP),intent(in) X) [private]

Definition at line 652 of file maths.f90.

7.112 MATHS::EIGENVALUE Interface Reference

Private Member Functions

- subroutine [EIGENVALUE_FULL_SP](#) (A, EVALUES, ERR, ERROR,*)
- subroutine [EIGENVALUE_FULL_DP](#) (A, EVALUES, ERR, ERROR,*)

7.112.1 Detailed Description

Definition at line 84 of file maths.f90.

7.112.2 Member Function Documentation

7.112.2.1 subroutine MATHS::EIGENVALUE::EIGENVALUE_FULL_DP
(REAL(DP),dimension(:,:),intent(in) *A*, REAL(DP),dimension(:,),intent(out) *EVALUES*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]

Definition at line 789 of file maths.f90.

7.112.2.2 subroutine MATHS::EIGENVALUE::EIGENVALUE_FULL_SP
(REAL(SP),dimension(:,:),intent(in) *A*, REAL(SP),dimension(:,),intent(out) *EVALUES*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]

Definition at line 697 of file maths.f90.

7.113 MATHS::EIGENVECTOR Interface Reference

Private Member Functions

- subroutine [EIGENVECTOR_FULL_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)
- subroutine [EIGENVECTOR_FULL_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)

7.113.1 Detailed Description

Definition at line 89 of file maths.f90.

7.113.2 Member Function Documentation

7.113.2.1 subroutine MATHS::EIGENVECTOR::EIGENVECTOR_FULL_DP
(REAL(DP),dimension(:,:),intent(in) *A*, REAL(DP),intent(in) *EVALUE*,
REAL(DP),dimension(:,intent(out) *EVECTOR*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 981 of file maths.f90.

7.113.2.2 subroutine MATHS::EIGENVECTOR::EIGENVECTOR_FULL_SP
(REAL(SP),dimension(:,:),intent(in) *A*, REAL(SP),intent(in) *EVALUE*,
REAL(SP),dimension(:,intent(out) *EVECTOR*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 890 of file maths.f90.

7.114 MATHS::I0 Interface Reference

Private Member Functions

- REAL(DP) [I0_DP](#) (X)
- REAL(SP) [I0_SP](#) (X)

7.114.1 Detailed Description

Definition at line 94 of file maths.f90.

7.114.2 Member Function Documentation

7.114.2.1 **REAL(DP) MATHS::I0::I0_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1081 of file maths.f90.

7.114.2.2 **REAL(SP) MATHS::I0::I0_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1116 of file maths.f90.

7.115 MATHS::I1 Interface Reference

Private Member Functions

- REAL(DP) [I1_DP](#) (X)
- REAL(SP) [I1_SP](#) (X)

7.115.1 Detailed Description

Definition at line 99 of file maths.f90.

7.115.2 Member Function Documentation

7.115.2.1 **REAL(DP) MATHS::I1::I1_DP (REAL(DP),intent(in) X) [private]**

Definition at line 1160 of file maths.f90.

7.115.2.2 **REAL(SP) MATHS::I1::I1_SP (REAL(SP),intent(in) X) [private]**

Definition at line 1195 of file maths.f90.

7.116 MATHS::INVERT Interface Reference

Private Member Functions

- subroutine [INVERT_FULL_SP](#) (A, B, DET, ERR, ERROR,*)
- subroutine [INVERT_FULL_DP](#) (A, B, DET, ERR, ERROR,*)

7.116.1 Detailed Description

Definition at line 104 of file maths.f90.

7.116.2 Member Function Documentation

7.116.2.1 subroutine MATHS::INVERT::INVERT_FULL_DP
(**REAL(DP),dimension(:,:),intent(in) A**, **REAL(DP),dimension(:,:),intent(out)**
B, **REAL(DP),intent(out) DET**, **INTEGER(INTG),intent(out) ERR**,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1316 of file maths.f90.

7.116.2.2 subroutine MATHS::INVERT::INVERT_FULL_SP
(**REAL(SP),dimension(:,:),intent(in) A**, **REAL(SP),dimension(:,:),intent(out)**
B, **REAL(SP),intent(out) DET**, **INTEGER(INTG),intent(out) ERR**,
TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Definition at line 1239 of file maths.f90.

7.117 MATHS::K0 Interface Reference

Private Member Functions

- REAL(DP) [K0_DP](#) (X)
- REAL(SP) [K0_SP](#) (X)

7.117.1 Detailed Description

Definition at line 109 of file maths.f90.

7.117.2 Member Function Documentation

7.117.2.1 REAL(DP) MATHS::K0::K0_DP (REAL(DP),intent(in) X) [private]

Definition at line 1402 of file maths.f90.

7.117.2.2 REAL(SP) MATHS::K0::K0_SP (REAL(SP),intent(in) X) [private]

Definition at line 1453 of file maths.f90.

7.118 MATHS::K1 Interface Reference

Private Member Functions

- REAL(DP) [K1_DP](#) (X)
- REAL(SP) [K1_SP](#) (X)

7.118.1 Detailed Description

Definition at line 114 of file maths.f90.

7.118.2 Member Function Documentation

7.118.2.1 REAL(DP) MATHS::K1::K1_DP (REAL(DP),intent(in) X) [private]

Definition at line 1513 of file maths.f90.

7.118.2.2 REAL(SP) MATHS::K1::K1_SP (REAL(SP),intent(in) X) [private]

Definition at line 1565 of file maths.f90.

7.119 MATHS::L2NORM Interface Reference

Private Member Functions

- REAL(SP) [L2NORM_SP](#) (A)
- REAL(DP) [L2NORM_DP](#) (A)

7.119.1 Detailed Description

Definition at line 119 of file maths.f90.

7.119.2 Member Function Documentation

7.119.2.1 REAL(DP) MATHS::L2NORM::L2NORM_DP (REAL(DP),dimension(:),intent(in) A) [private]

Definition at line 1735 of file maths.f90.

7.119.2.2 REAL(SP) MATHS::L2NORM::L2NORM_SP (REAL(SP),dimension(:),intent(in) A) [private]

Definition at line 1711 of file maths.f90.

7.120 MATHS::NORMALISE Interface Reference

Private Member Functions

- REAL(SP) [NORMALISE_SP](#) (A, ERR, ERROR)
- REAL(DP) [NORMALISE_DP](#) (A, ERR, ERROR)

7.120.1 Detailed Description

Definition at line 124 of file maths.f90.

7.120.2 Member Function Documentation

7.120.2.1 REAL(DP) MATHS::NORMALISE::NORMALISE_DP (REAL(DP),dimension(:),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 1805 of file maths.f90.

7.120.2.2 REAL(SP) MATHS::NORMALISE::NORMALISE_SP (REAL(SP),dimension(:),intent(in) A, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Definition at line 1768 of file maths.f90.

7.121 MATHS::SOLVE_SMALL_LINEAR_SYSTEM Interface Reference

Private Member Functions

- subroutine `SOLVE_SMALL_LINEAR_SYSTEM_SP` (*A*, *x*, *b*, *ERR*, *ERROR*,*)
- subroutine `SOLVE_SMALL_LINEAR_SYSTEM_DP` (*A*, *x*, *b*, *ERR*, *ERROR*,*)

7.121.1 Detailed Description

Definition at line 129 of file maths.f90.

7.121.2 Member Function Documentation

**7.121.2.1 subroutine MATHS::SOLVE_SMALL_LINEAR_SYSTEM::SOLVE_-
SMALL_LINEAR_SYSTEM_DP (REAL(DP),dimension(:,:),intent(in) *A*,
REAL(DP),dimension(:,intent(out) *x*, REAL(DP),dimension(:,intent(in) *b*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Definition at line 1900 of file maths.f90.

**7.121.2.2 subroutine MATHS::SOLVE_SMALL_LINEAR_SYSTEM::SOLVE_-
SMALL_LINEAR_SYSTEM_SP (REAL(SP),dimension(:,:),intent(in) *A*,
REAL(SP),dimension(:,intent(out) *x*, REAL(SP),dimension(:,intent(in) *b*,
INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*,
*) [private]**

Definition at line 1851 of file maths.f90.

7.122 MATRIX_VECTOR::MATRIX_ALL_VALUES_SET Interface Reference

Public Member Functions

- subroutine [MATRIX_ALL_VALUES_SET_INTG](#) (MATRIX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_ALL_VALUES_SET_SP](#) (MATRIX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_ALL_VALUES_SET_DP](#) (MATRIX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_ALL_VALUES_SET_L](#) (MATRIX, VALUE, ERR, ERROR,*)

7.122.1 Detailed Description

Definition at line 178 of file matrix_vector.f90.

7.122.2 Member Function Documentation

7.122.2.1 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET::MATRIX_ALL_VALUES_SET_DP (TYPE(MATRIX_TYPE),pointer *MATRIX*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 373 of file matrix_vector.f90.

7.122.2.2 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET::MATRIX_ALL_VALUES_SET_INTG (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 293 of file matrix_vector.f90.

7.122.2.3 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET::MATRIX_ALL_VALUES_SET_L (TYPE(MATRIX_TYPE),pointer *MATRIX*, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 413 of file matrix_vector.f90.

7.122.2.4 subroutine MATRIX_VECTOR::MATRIX_ALL_VALUES_SET::MATRIX_ALL_VALUES_SET_SP (TYPE(MATRIX_TYPE),pointer *MATRIX*, REAL(SP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 333 of file matrix_vector.f90.

7.123 MATRIX_VECTOR::MATRIX_DATA_GET Interface Reference

Public Member Functions

- subroutine [MATRIX_DATA_GET_INTG](#) (MATRIX, DATA, ERR, ERROR,*)
- subroutine [MATRIX_DATA_GET_SP](#) (MATRIX, DATA, ERR, ERROR,*)
- subroutine [MATRIX_DATA_GET_DP](#) (MATRIX, DATA, ERR, ERROR,*)
- subroutine [MATRIX_DATA_GET_L](#) (MATRIX, DATA, ERR, ERROR,*)

7.123.1 Detailed Description

Definition at line 185 of file matrix_vector.f90.

7.123.2 Member Function Documentation

7.123.2.1 subroutine MATRIX_VECTOR::MATRIX_DATA_GET::MATRIX_DATA_GET_DP
`(TYPE(MATRIX_TYPE),pointer MATRIX, REAL(DP),dimension(:),pointer DATA,
 INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
 *)`

Parameters:

MATRIX A pointer to the matrix

DATA On return a pointer to the matrix data

ERR The error code

ERROR The error string

Definition at line 712 of file matrix_vector.f90.

7.123.2.2 subroutine MATRIX_VECTOR::MATRIX_DATA_GET::MATRIX_DATA_GET_INTG
`(TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),pointer
DATA, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out)
ERROR, *)`

Parameters:

MATRIX A pointer to the matrix

DATA On return a pointer to the matrix data

ERR The error code

ERROR The error string

Definition at line 622 of file matrix_vector.f90.

7.123.2.3 subroutine MATRIX_VECTOR::MATRIX_DATA_GET::MATRIX_DATA_GET_L
(**TYPE(MATRIX_TYPE)**,pointer **MATRIX**, **LOGICAL**,dimension(:),pointer **DATA**,
INTEGER(INTG),intent(out) **ERR**, **TYPE(VARYING_STRING)**,intent(out) **ERROR**,
*)

Parameters:

MATRIX A pointer to the matrix
DATA On return a pointer to the matrix data
ERR The error code
ERROR The error string

Definition at line 757 of file matrix_vector.f90.

7.123.2.4 subroutine MATRIX_VECTOR::MATRIX_DATA_GET::MATRIX_DATA_GET_SP
(**TYPE(MATRIX_TYPE)**,pointer **MATRIX**, **REAL(SP)**,dimension(:),pointer **DATA**,
INTEGER(INTG),intent(out) **ERR**, **TYPE(VARYING_STRING)**,intent(out) **ERROR**,
*)

Parameters:

MATRIX A pointer to the matrix
DATA On return a pointer to the matrix data
ERR The error code
ERROR The error string

Definition at line 667 of file matrix_vector.f90.

7.124 MATRIX_VECTOR::MATRIX_VALUES_ADD Interface Reference

Public Member Functions

- subroutine `MATRIX_VALUES_ADD_INTG` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_INTG1` (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_INTG2` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_SP` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_SP1` (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_SP2` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_DP` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_DP1` (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_DP2` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_L` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_L1` (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine `MATRIX_VALUES_ADD_L2` (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

7.124.1 Detailed Description

Definition at line 192 of file matrix_vector.f90.

7.124.2 Member Function Documentation

7.124.2.1 subroutine `MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_DP` (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(DP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES *ROW_INDICES*(i). The row index for the i'th value to add

COLUMN_INDICES *COLUMN_INDICES*(i). The column index for the i'th value to add

VALUES *VALUES*(i). The value of the i'th value to add

ERR The error code

ERROR The error string

Definition at line 2166 of file matrix_vector.f90.

7.124.2.2 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_DP1 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW_INDEX*, INTEGER(INTG),intent(in) *COLUMN_INDEX*, REAL(DP),intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index for the value to add

COLUMN_INDEX The column index for the value to add

VALUE The value to add

ERR The error code

ERROR The error string

Definition at line 2232 of file matrix_vector.f90.

7.124.2.3 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_DP2 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*, REAL(DP),dimension(:, :,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to add

COLUMN_INDICES COLUMN_INDICES(j). The column index for the ij'th value to add

VALUES VALUES(i,j). The value of the ij'th value to add

ERR The error code

ERROR The error string

Definition at line 2282 of file matrix_vector.f90.

7.124.2.4 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_INTG (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*, INTEGER(INTG),dimension(:,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to add
COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to add
VALUES VALUES(i). The value of the i'th value to add
ERR The error code
ERROR The error string

Definition at line 1794 of file matrix_vector.f90.

7.124.2.5 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_INTG1 (TYPE(MATRIX_TYPE),pointer **MATRIX**,
 INTEGER(INTG),intent(in) **ROW_INDEX**, INTEGER(INTG),intent(in)
COLUMN_INDEX, INTEGER(INTG),intent(in) **VALUE**, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index for the value to add
COLUMN_INDEX The column index for the value to add
VALUE The value to add
ERR The error code
ERROR The error string

Definition at line 1860 of file matrix_vector.f90.

7.124.2.6 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_INTG2 (TYPE(MATRIX_TYPE),pointer **MATRIX**,
 INTEGER(INTG),dimension(:),intent(in) **ROW_INDICES**,
 INTEGER(INTG),dimension(:),intent(in) **COLUMN_INDICES**,
 INTEGER(INTG),dimension(:, :,),intent(in) **VALUES**, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to add
COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to add
VALUES VALUES(i,j). The value of the ij'th value to add
ERR The error code
ERROR The error string

Definition at line 1910 of file matrix_vector.f90.

7.124.2.7 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_L (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*, LOGICAL,dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDICES*** *ROW_INDICES*(i). The row index for the i'th value to add***COLUMN_INDICES*** *COLUMN_INIDICES*(i). The column index for the i'th value to add***VALUES*** *VALUES*(i). The value of the i'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2352 of file matrix_vector.f90.

7.124.2.8 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_L1 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW_INDEX*, INTEGER(INTG),intent(in) *COLUMN_INDEX*, LOGICAL,intent(in) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDEX*** The row index for the value to add***COLUMN_INDEX*** The column index for the value to add***VALUE*** The value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2418 of file matrix_vector.f90.

7.124.2.9 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_L2 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*, LOGICAL,dimension(:, :,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDICES*** *ROW_INDICES*(i). The row index for the ij'th value to add***COLUMN_INDICES*** *COLUMN_INIDICES*(j). The column index for the ij'th value to add***VALUES*** *VALUES*(i,j). The value of the ij'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2468 of file matrix_vector.f90.

7.124.2.10 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_SP (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(SP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDICES*** *ROW_INDICES*(i). The row index for the i'th value to add***COLUMN_INDICES*** *COLUMN_INIDICES*(i). The column index for the i'th value to add***VALUES*** *VALUES*(i). The value of the i'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 1980 of file matrix_vector.f90.

7.124.2.11 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_SP1 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)
ROW_INDEX, INTEGER(INTG),intent(in) *COLUMN_INDEX*, REAL(SP),intent(in) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDEX*** The row index for the value to add***COLUMN_INDEX*** The column index for the value to add***VALUE*** The value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2046 of file matrix_vector.f90.

7.124.2.12 subroutine MATRIX_VECTOR::MATRIX_VALUES_ADD::MATRIX_VALUES_ADD_SP2 (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(SP),dimension(:, :),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDICES*** *ROW_INDICES*(i). The row index for the ij'th value to add***COLUMN_INDICES*** *COLUMN_INIDICES*(j). The column index for the ij'th value to add***VALUES*** *VALUES*(i,j). The value of the ij'th value to add***ERR*** The error code***ERROR*** The error string

Definition at line 2096 of file matrix_vector.f90.

7.125 MATRIX_VECTOR::MATRIX_VALUES_GET Interface Reference

Public Member Functions

- subroutine [MATRIX_VALUES_GET_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_DP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_DP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_DP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_L](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_L1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_GET_L2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

7.125.1 Detailed Description

Definition at line 207 of file matrix_vector.f90.

7.125.2 Member Function Documentation

7.125.2.1 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_DP (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(DP),dimension(:),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES *ROW_INDICES*(i). The row index for the i'th value to get

COLUMN_INDICES *COLUMN_INDICES*(i). The column index for the i'th value to get

VALUES *VALUES*(i). On return the value of the i'th value to get

ERR The error code

ERROR The error string

Definition at line 2898 of file matrix_vector.f90.

**7.125.2.2 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_-
GET_DP1** (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)
ROW_INDEX, INTEGER(INTG),intent(in) *COLUMN_INDEX*, REAL(DP),intent(out)
VALUE, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out)
ERROR, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to get

COLUMN_INDEX The column index of the value to get

VALUE On return the value in the matrix at the specified row and column

ERR The error code

ERROR The error string

Definition at line 2962 of file matrix_vector.f90.

**7.125.2.3 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_-
VALUES_GET_DP2** (TYPE(MATRIX_TYPE),pointer *MATRIX*,
INTEGER(INTG),dimension(:,),intent(in) *ROW_INDICES*,
INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*,
REAL(DP),dimension(:, :,),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to get

COLUMN_INDICES COLUMN_INDICES(j). The column index for the ij'th value to get

VALUES VALUES(i,j). On return the value of the ij'th value to get

ERR The error code

ERROR The error string

Definition at line 3010 of file matrix_vector.f90.

**7.125.2.4 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_-
VALUES_GET_INTG** (TYPE(MATRIX_TYPE),pointer *MATRIX*,
INTEGER(INTG),dimension(:,),intent(in) *ROW_INDICES*,
INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*,
INTEGER(INTG),dimension(:,),intent(out) *VALUES*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to get

COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to get

VALUES VALUES(i). On return the value of the i'th value to get

ERR The error code

ERROR The error string

Definition at line 2538 of file matrix_vector.f90.

7.125.2.5 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_INTG1 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to get

COLUMN_INDEX The column index of the value to get

VALUE On return the value in the matrix at the specified row and column

ERR The error code

ERROR The error string

Definition at line 2602 of file matrix_vector.f90.

7.125.2.6 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_INTG2 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:,),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) COLUMN_INDICES, INTEGER(INTG),dimension(:,:,),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to get

COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to get

VALUES VALUES(i,j). On return the value of the ij'th value to get

ERR The error code

ERROR The error string

Definition at line 2650 of file matrix_vector.f90.

7.125.2.7 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_L (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:**MATRIX** A pointer to the matrix**ROW_INDICES** ROW_INDICES(i). The row index for the i'th value to get**COLUMN_INDICES** COLUMN_INIDICES(i). The column index for the i'th value to get**VALUES** VALUES(i). On return the value of the i'th value to get**ERR** The error code**ERROR** The error string

Definition at line 3078 of file matrix_vector.f90.

7.125.2.8 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_L1 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),intent(in) ROW_INDEX, INTEGER(INTG),intent(in) COLUMN_INDEX, LOGICAL,intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:**MATRIX** A pointer to the matrix**ROW_INDEX** The row index of the value to get**COLUMN_INDEX** The column index of the value to get**VALUE** On return the value in the matrix at the specified row and column**ERR** The error code**ERROR** The error string

Definition at line 3142 of file matrix_vector.f90.

7.125.2.9 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_L2 (TYPE(MATRIX_TYPE),pointer MATRIX, INTEGER(INTG),dimension(:),intent(in) ROW_INDICES, INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES, LOGICAL,dimension(:, :, intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:**MATRIX** A pointer to the matrix**ROW_INDICES** ROW_INDICES(i). The row index for the ij'th value to get**COLUMN_INDICES** COLUMN_INIDICES(j). The column index for the ij'th value to get**VALUES** VALUES(i,j). On return the value of the ij'th value to get**ERR** The error code**ERROR** The error string

Definition at line 3190 of file matrix_vector.f90.

7.125.2.10 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_SP (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*, REAL(SP),dimension(:),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES *ROW_INDICES*(i). The row index for the i'th value to get

COLUMN_INDICES *COLUMN_INIDICES*(i). The column index for the i'th value to get

VALUES *VALUES*(i). On return the value of the i'th value to get

ERR The error code

ERROR The error string

Definition at line 2718 of file matrix_vector.f90.

7.125.2.11 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_SP1 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in) *ROW_INDEX*, INTEGER(INTG),intent(in) *COLUMN_INDEX*, REAL(SP),intent(out) *VALUE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to get

COLUMN_INDEX The column index of the value to get

VALUE On return the value in the matrix at the specified row and column

ERR The error code

ERROR The error string

Definition at line 2782 of file matrix_vector.f90.

7.125.2.12 subroutine MATRIX_VECTOR::MATRIX_VALUES_GET::MATRIX_VALUES_GET_SP2 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*, INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*, REAL(SP),dimension(:, :),intent(out) *VALUES*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES *ROW_INDICES*(i). The row index for the ij'th value to get

COLUMN_INDICES *COLUMN_INIDICES*(j). The column index for the ij'th value to get

VALUES *VALUES*(i,j). On return the value of the ij'th value to get

ERR The error code

ERROR The error string

Definition at line 2830 of file matrix_vector.f90.

7.126 MATRIX_VECTOR::MATRIX_VALUES_SET Interface Reference

Public Member Functions

- subroutine [MATRIX_VALUES_SET_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_DP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_DP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_DP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_L](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_L1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)
- subroutine [MATRIX_VALUES_SET_L2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

7.126.1 Detailed Description

Definition at line 222 of file matrix_vector.f90.

7.126.2 Member Function Documentation

7.126.2.1 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_DP (TYPE(MATRIX_TYPE),pointer *MATRIX*,
INTEGER(INTG),dimension(:),intent(in) ROW_INDICES,
INTEGER(INTG),dimension(:),intent(in) COLUMN_INDICES,
REAL(DP),dimension(:),intent(in) VALUES, *INTEGER(INTG),intent(out) ERR*,
*TYPE(VARYING_STRING),intent(out) ERROR, **)

Parameters:

MATRIX A pointer to the matrix to set.

ROW_INDICES *ROW_INDICES(i)*. The row index of the i'th value to set

COLUMN_INDICES *COLUMN_INDICES(i)*. The column index of the i'th value to set

VALUES *VALUES(i)*. The value of the i'th value to set

ERR The error code

ERROR The error string

Definition at line 3630 of file matrix_vector.f90.

7.126.2.2 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_DP1 (TYPE(MATRIX_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **ROW_INDEX**, INTEGER(INTG),intent(in) **COLUMN_INDEX**, REAL(DP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3696 of file matrix_vector.f90.

7.126.2.3 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_DP2 (TYPE(MATRIX_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN_INDICES**, REAL(DP),dimension(:, :,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix to set.

ROW_INDICES ROW_INDICES(i). The row index of the ij'th value to set

COLUMN_INDICES COLUMN_INDICES(j). The column index of the ij'th value to set

VALUES VALUES(i,j). The value of the ij'th value to set

ERR The error code

ERROR The error string

Definition at line 3746 of file matrix_vector.f90.

7.126.2.4 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_INTG (TYPE(MATRIX_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDICES ROW_INDICES(i). The row index for the i'th value to set
COLUMN_INDICES COLUMN_INIDICES(i). The column index for the i'th value to set
VALUES VALUES(i). The value of the i'th value to set
ERR The error code
ERROR The error string

Definition at line 3258 of file matrix_vector.f90.

7.126.2.5 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_INTG1 (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),intent(in) *ROW_INDEX*, INTEGER(INTG),intent(in)
COLUMN_INDEX, INTEGER(INTG),intent(in) *VALUE*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix
ROW_INDEX The row index of the value to set
COLUMN_INDEX The column index of the value to set
VALUE The value to set
ERR The error code
ERROR The error string

Definition at line 3324 of file matrix_vector.f90.

7.126.2.6 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_INTG2 (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:,),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*,
 INTEGER(INTG),dimension(:,:,),intent(in) *VALUES*, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix
ROW_INDICES ROW_INDICES(i). The row index for the ij'th value to set
COLUMN_INDICES COLUMN_INIDICES(j). The column index for the ij'th value to set
VALUES VALUES(i,j). The value of the i,j'th value to set
ERR The error code
ERROR The error string

Definition at line 3374 of file matrix_vector.f90.

7.126.2.7 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_L (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),dimension(:,),intent(in)
ROW_INDICES, INTEGER(INTG),dimension(:,),intent(in) *COLUMN_INDICES*,
 LOGICAL,dimension(:,),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES ROW_INDICES(i). The row index of the i'th value to set

COLUMN_INDICES COLUMN_INDICES(i). The column index of the i'th value to set

VALUES VALUES(i). The value of the i'th value to set

ERR The error code

ERROR The error string

Definition at line 3816 of file matrix_vector.f90.

7.126.2.8 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_L1 (TYPE(MATRIX_TYPE),pointer **MATRIX**, INTEGER(INTG),intent(in) **ROW_INDEX**, INTEGER(INTG),intent(in) **COLUMN_INDEX**, LOGICAL,intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix

ROW_INDEX The row index of the value to set

COLUMN_INDEX The column index of the value to set

VALUE The value to set

ERR The error code

ERROR The error string

Definition at line 3882 of file matrix_vector.f90.

7.126.2.9 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_L2 (TYPE(MATRIX_TYPE),pointer **MATRIX**, INTEGER(INTG),dimension(:,),intent(in) **ROW_INDICES**, INTEGER(INTG),dimension(:,),intent(in) **COLUMN_INDICES**, LOGICAL,dimension(:, :,),intent(in) **VALUES**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

MATRIX A pointer to the matrix to set

ROW_INDICES ROW_INDICES(i). The row index of the ij'th value to set

COLUMN_INDICES COLUMN_INDICES(j). The column index of the ij'th value to set

VALUES VALUES(i,j). The value of the ij'th value to set

ERR The error code

ERROR The error string

Definition at line 3932 of file matrix_vector.f90.

7.126.2.10 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_SP (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(SP),dimension(:),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix to set***ROW_INDICES*** *ROW_INDICES*(i). The row index of the i'th value to set***COLUMN_INDICES*** *COLUMN_INDICES*(i). The column index of the i'th value to set***VALUES*** *VALUES*(i). The value of the i'th value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3444 of file matrix_vector.f90.

7.126.2.11 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_SP1 (TYPE(MATRIX_TYPE),pointer *MATRIX*, INTEGER(INTG),intent(in)
ROW_INDEX, INTEGER(INTG),intent(in) *COLUMN_INDEX*, REAL(SP),intent(in) *VALUE*,
 INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix***ROW_INDEX*** The row index of the value to set***COLUMN_INDEX*** The column index of the value to set***VALUE*** The value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3510 of file matrix_vector.f90.

7.126.2.12 subroutine MATRIX_VECTOR::MATRIX_VALUES_SET::MATRIX_VALUES_SET_SP2 (TYPE(MATRIX_TYPE),pointer *MATRIX*,
 INTEGER(INTG),dimension(:),intent(in) *ROW_INDICES*,
 INTEGER(INTG),dimension(:),intent(in) *COLUMN_INDICES*,
 REAL(SP),dimension(:, :),intent(in) *VALUES*, INTEGER(INTG),intent(out) *ERR*,
 TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:***MATRIX*** A pointer to the matrix to set***ROW_INDICES*** *ROW_INDICES*(i). The row index of the ij'th value to set***COLUMN_INDICES*** *COLUMN_INDICES*(j). The column index of the ij'th value to set***VALUES*** *VALUES*(i,j). The value of the ij'th value to set***ERR*** The error code***ERROR*** The error string

Definition at line 3560 of file matrix_vector.f90.

7.127 MATRIX_VECTOR::VECTOR_ALL_VALUES_SET Interface Reference

Public Member Functions

- subroutine [VECTOR_ALL_VALUES_SET_INTG](#) (VECTOR, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_ALL_VALUES_SET_SP](#) (VECTOR, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_ALL_VALUES_SET_DP](#) (VECTOR, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_ALL_VALUES_SET_L](#) (VECTOR, VALUE, ERR, ERROR,*)

7.127.1 Detailed Description

Definition at line 237 of file matrix_vector.f90.

7.127.2 Member Function Documentation

7.127.2.1 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET::VECTOR_ALL_VALUES_SET_DP (TYPE(VECTOR_TYPE),pointer VECTOR, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector to set

VALUE The value to set the vector to

ERR The error code

ERROR The error string

Definition at line 4082 of file matrix_vector.f90.

7.127.2.2 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET::VECTOR_ALL_VALUES_SET_INTG (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector to set

VALUE The value to set the vector to

ERR The error code

ERROR The error string

Definition at line 4002 of file matrix_vector.f90.

7.127.2.3 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET::VECTOR_ALL_VALUES_SET_L (TYPE(VECTOR_TYPE),pointer **VECTOR**, LOGICAL,intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

VECTOR A pointer to the vector to set
VALUE The value to set the vector to
ERR The error code
ERROR The error string

Definition at line 4122 of file matrix_vector.f90.

7.127.2.4 subroutine MATRIX_VECTOR::VECTOR_ALL_VALUES_SET::VECTOR_ALL_VALUES_SET_SP (TYPE(VECTOR_TYPE),pointer **VECTOR**, REAL(SP),intent(in) **VALUE**, INTEGER(INTG),intent(out) **ERR**, TYPE(VARYING_STRING),intent(out) **ERROR**, *)

Parameters:

VECTOR A pointer to the vector to set
VALUE The value to set the vector to
ERR The error code
ERROR The error string

Definition at line 4042 of file matrix_vector.f90.

7.128 MATRIX_VECTOR::VECTOR_DATA_GET Interface Reference

Public Member Functions

- subroutine [VECTOR_DATA_GET_INTG](#) (VECTOR, DATA, ERR, ERROR,*)
- subroutine [VECTOR_DATA_GET_SP](#) (VECTOR, DATA, ERR, ERROR,*)
- subroutine [VECTOR_DATA_GET_DP](#) (VECTOR, DATA, ERR, ERROR,*)
- subroutine [VECTOR_DATA_GET_L](#) (VECTOR, DATA, ERR, ERROR,*)

7.128.1 Detailed Description

Definition at line 244 of file matrix_vector.f90.

7.128.2 Member Function Documentation

7.128.2.1 subroutine MATRIX_VECTOR::VECTOR_DATA_GET::VECTOR_DATA_GET_DP (TYPE(VECTOR_TYPE),pointer *VECTOR*, REAL(DP),dimension(:),pointer *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

VECTOR A pointer to the vector

DATA On return a pointer to the vector data

ERR The error code

ERROR The error string

Definition at line 4339 of file matrix_vector.f90.

7.128.2.2 subroutine MATRIX_VECTOR::VECTOR_DATA_GET::VECTOR_DATA_GET_INTG (TYPE(VECTOR_TYPE),pointer *VECTOR*, INTEGER(INTG),dimension(:),pointer *DATA*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *)

Parameters:

VECTOR A pointer to the vector

DATA On return a pointer to the vector data

ERR The error code

ERROR The error string

Definition at line 4249 of file matrix_vector.f90.

7.128.2.3 subroutine MATRIX_VECTOR::VECTOR_DATA_GET::VECTOR_DATA_GET_L
(**TYPE(VECTOR_TYPE),pointer VECTOR, LOGICAL,dimension(:),pointer DATA,**
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***)**

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4384 of file matrix_vector.f90.

7.128.2.4 subroutine MATRIX_VECTOR::VECTOR_DATA_GET::VECTOR_DATA_GET_SP
(**TYPE(VECTOR_TYPE),pointer VECTOR, REAL(SP),dimension(:),pointer DATA,**
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
***)**

Parameters:

VECTOR A pointer to the vector
DATA On return a pointer to the vector data
ERR The error code
ERROR The error string

Definition at line 4294 of file matrix_vector.f90.

7.129 MATRIX_VECTOR::VECTOR_VALUES_GET Interface Reference

Public Member Functions

- subroutine [VECTOR_VALUES_GET_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine [VECTOR_VALUES_GET_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

7.129.1 Detailed Description

Definition at line 251 of file matrix_vector.f90.

7.129.2 Member Function Documentation

**7.129.2.1 subroutine MATRIX_VECTOR::VECTOR_VALUES_-
GET::VECTOR_VALUES_GET_DP (TYPE(VECTOR_TYPE),pointer
VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES,
REAL(DP),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4853 of file matrix_vector.f90.

**7.129.2.2 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_-
GET_DP1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in)
INDEX, REAL(DP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR, *)**

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 4911 of file matrix_vector.f90.

7.129.2.3 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_INTG (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4643 of file matrix_vector.f90.

7.129.2.4 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_INTG1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 4701 of file matrix_vector.f90.

7.129.2.5 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_L (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4958 of file matrix_vector.f90.

7.129.2.6 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_L1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 5016 of file matrix_vector.f90.

7.129.2.7 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_SP (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:,intent(out) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to get
VALUES VALUES(i). On return the i'th value to get
ERR The error code
ERROR The error string

Definition at line 4748 of file matrix_vector.f90.

7.129.2.8 subroutine MATRIX_VECTOR::VECTOR_VALUES_GET::VECTOR_VALUES_GET_SP1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(SP),intent(out) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index of the vector to get
VALUE On return the value of the vector at the specified index
ERR The error code
ERROR The error string

Definition at line 4806 of file matrix_vector.f90.

7.130 MATRIX_VECTOR::VECTOR_VALUES_SET Interface Reference

Public Member Functions

- subroutine `VECTOR_VALUES_SET_INTG` (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_INTG1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_SP` (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_SP1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_DP` (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_DP1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_L` (VECTOR, INDICES, VALUES, ERR, ERROR,*)
- subroutine `VECTOR_VALUES_SET_L1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)

7.130.1 Detailed Description

Definition at line 262 of file matrix_vector.f90.

7.130.2 Member Function Documentation

7.130.2.1 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_DP (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(DP),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5273 of file matrix_vector.f90.

7.130.2.2 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_DP1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(DP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index to set
VALUE The value to set at the specified index
ERR The error code
ERROR The error string

Definition at line 5331 of file matrix_vector.f90.

7.130.2.3 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_INTG (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, INTEGER(INTG),dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5063 of file matrix_vector.f90.

7.130.2.4 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_INTG1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, INTEGER(INTG),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index to set
VALUE The value to set at the specified index
ERR The error code
ERROR The error string

Definition at line 5121 of file matrix_vector.f90.

7.130.2.5 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_L (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, LOGICAL,dimension(:),intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5378 of file matrix_vector.f90.

7.130.2.6 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_L1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, LOGICAL,intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index to set
VALUE The value to set at the specified index
ERR The error code
ERROR The error string

Definition at line 5436 of file matrix_vector.f90.

7.130.2.7 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_SP (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),dimension(:),intent(in) INDICES, REAL(SP),dimension(:,intent(in) VALUES, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDICES INDICES(i). The i'th index to set
VALUES VALUES(i). The i'th value to set
ERR The error code
ERROR The error string

Definition at line 5168 of file matrix_vector.f90.

7.130.2.8 subroutine MATRIX_VECTOR::VECTOR_VALUES_SET::VECTOR_VALUES_SET_SP1 (TYPE(VECTOR_TYPE),pointer VECTOR, INTEGER(INTG),intent(in) INDEX, REAL(SP),intent(in) VALUE, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *)

Parameters:

VECTOR A pointer to the vector
INDEX The index to set
VALUE The value to set at the specified index
ERR The error code
ERROR The error string

Definition at line 5226 of file matrix_vector.f90.

7.131 MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET Interface Reference

Sets/changes the number of mesh components for a mesh.

Private Member Functions

- subroutine `MESH_NUMBER_OF_COMPONENTS_SET_NUMBER` (`USER_NUMBER`, `REGION`, `NUMBER_OF_COMPONENTS`, `ERR`, `ERROR,*`)
- subroutine `MESH_NUMBER_OF_COMPONENTS_SET_PTR` (`MESH`, `NUMBER_OF_COMPONENTS`, `ERR`, `ERROR,*`)

7.131.1 Detailed Description

Sets/changes the number of mesh components for a mesh.

Definition at line 85 of file mesh_routines.f90.

7.131.2 Member Function Documentation

7.131.2.1 subroutine MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET::MESH_NUMBER_OF_COMPONENTS_SET_NUMBER
`(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer REGION, INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Parameters:

`USER_NUMBER` The user number of the mesh to set the number of components for
`REGION` A pointer to the region containing the mesh

`NUMBER_OF_COMPONENTS` The number of components to set for the mesh

`ERR` The error code

`ERROR` The error string

Definition at line 4438 of file mesh_routines.f90.

7.131.2.2 subroutine MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET::MESH_NUMBER_OF_COMPONENTS_SET_PTR
`(TYPE(MESH_TYPE),pointer MESH, INTEGER(INTG),intent(in) NUMBER_OF_COMPONENTS, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]`

Parameters:

`MESH` A pointer to the mesh to set the number of components for

`NUMBER_OF_COMPONENTS` The number of components to set.

`ERR` The error code

`ERROR` The error string

Definition at line 4476 of file mesh_routines.f90.

7.132 MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_- SET Interface Reference

Sets/changes the number of elements for a mesh.

Private Member Functions

- subroutine [MESH_NUMBER_OF_ELEMENTS_SET_NUMBER](#) (*USER_NUMBER*, *REGION*, *NUMBER_OF_ELEMENTS*, *ERR*, *ERROR*,*)
- subroutine [MESH_NUMBER_OF_ELEMENTS_SET_PTR](#) (*MESH*, *NUMBER_OF_ELEMENTS*, *ERR*, *ERROR*,*)

7.132.1 Detailed Description

Sets/changes the number of elements for a mesh.

Definition at line 91 of file mesh_routines.f90.

7.132.2 Member Function Documentation

**7.132.2.1 subroutine MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_-
SET::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER**
*(INTEGER(INTG),intent(in) USER_NUMBER, TYPE(REGION_TYPE),pointer
REGION, INTEGER(INTG),intent(in) NUMBER_OF_ELEMENTS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
) [private]

Parameters:

USER_NUMBER The user number of the mesh to set the number of elements for
REGION A pointer to the region containing the mesh
NUMBER_OF_ELEMENTS The number of elements to set
ERR The error code
ERROR The error string

Definition at line 4546 of file mesh_routines.f90.

**7.132.2.2 subroutine MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_-
SET::MESH_NUMBER_OF_ELEMENTS_SET_PTR** (*TYPE(MESH_TYPE),pointer
MESH, INTEGER(INTG),intent(in) NUMBER_OF_ELEMENTS,
INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR,
) [private])

Parameters:

MESH A pointer to the mesh to set the number of elements for
NUMBER_OF_ELEMENTS The number of elements to set
ERR The error code
ERROR The error string

Definition at line 4584 of file mesh_routines.f90.

7.133 PROBLEM_ROUTINES::PROBLEM_DESTROY Interface Reference

Private Member Functions

- subroutine [PROBLEM_DESTROY_NUMBER](#) (USER_NUMBER, ERR, ERROR,*)
- subroutine [PROBLEM_DESTROY_PTR](#) (PROBLEM, ERR, ERROR,*)

7.133.1 Detailed Description

Definition at line 97 of file problem_routines.f90.

7.133.2 Member Function Documentation

7.133.2.1 subroutine PROBLEM_ROUTINES::PROBLEM_DESTROY::PROBLEM_DESTROY_NUMBER (INTEGER(INTG),intent(in) USER_NUMBER, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

USER_NUMBER The user number of the problem to destroy

ERR The error code

ERROR The error string

Definition at line 253 of file problem_routines.f90.

7.133.2.2 subroutine PROBLEM_ROUTINES::PROBLEM_DESTROY::PROBLEM_DESTROY_PTR (TYPE(PROBLEM_TYPE),pointer PROBLEM, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, *) [private]

Parameters:

PROBLEM A pointer to the problem to destroy

ERR The error code

ERROR The error string

Definition at line 304 of file problem_routines.f90.

7.134 PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET Interface Reference

Private Member Functions

- subroutine **PROBLEM_SPECIFICATION_SET_NUMBER** (*USER_NUMBER*, *PROBLEM_CLASS*, *PROBLEM_EQUATION_TYPE*, *PROBLEM_SUBTYPE*, *ERR*, *ERROR*,*)
- subroutine **PROBLEM_SPECIFICATION_SET_PTR** (*PROBLEM*, *PROBLEM_CLASS*, *PROBLEM_EQUATION_TYPE*, *PROBLEM_SUBTYPE*, *ERR*, *ERROR*,*)

7.134.1 Detailed Description

Definition at line 102 of file problem_routines.f90.

7.134.2 Member Function Documentation

7.134.2.1 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET::PROBLEM_SPECIFICATION_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*, INTEGER(INTG),intent(in) *PROBLEM_CLASS*, INTEGER(INTG),intent(in) *PROBLEM_EQUATION_TYPE*, INTEGER(INTG),intent(in) *PROBLEM_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

USER_NUMBER The user number of the problem to set the specification for.

PROBLEM_CLASS The problem class to set.

PROBLEM_EQUATION_TYPE The problem equation to set.

PROBLEM_SUBTYPE The problem subtype.

ERR The error code

ERROR The error string

Definition at line 1208 of file problem_routines.f90.

7.134.2.2 subroutine PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET::PROBLEM_SPECIFICATION_SET_PTR (TYPE(*PROBLEM_TYPE*),pointer *PROBLEM*, INTEGER(INTG),intent(in) *PROBLEM_CLASS*, INTEGER(INTG),intent(in) *PROBLEM_EQUATION_TYPE*, INTEGER(INTG),intent(in) *PROBLEM_SUBTYPE*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Parameters:

PROBLEM A pointer to the problem to set the specification for.

PROBLEM_CLASS The problem class to set.

PROBLEM_EQUATION_TYPE The problem equation type to set.

PROBLEM_SUBTYPE The problem subtype to set.

ERR The error code

ERROR The error string

Definition at line 1237 of file problem_routines.f90.

7.135 REGION_ROUTINES::REGION_COORDINATE_- SYSTEM_SET Interface Reference

Private Member Functions

- subroutine [REGION_COORDINATE_SYSTEM_SET_NUMBER](#) (USER_NUMBER,
COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine [REGION_COORDINATE_SYSTEM_SET_PTR](#) (REGION, COORDINATE_SYSTEM,
ERR, ERROR,*)

7.135.1 Detailed Description

Definition at line 71 of file region_routines.f90.

7.135.2 Member Function Documentation

7.135.2.1 subroutine **REGION_ROUTINES::REGION_COORDINATE_-
SYSTEM_SET:::REGION_COORDINATE_SYSTEM_SET_NUMBER**
(INTEGER(INTG),intent(in) *USER_NUMBER*, TYPE(COORDINATE_SYSTEM_-
TYPE),pointer *COORDINATE_SYSTEM*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 135 of file region_routines.f90.

7.135.2.2 subroutine **REGION_ROUTINES::REGION_COORDINATE_-
SYSTEM_SET:::REGION_COORDINATE_SYSTEM_SET_PTR**
(TYPE(REGION_TYPE),pointer *REGION*, TYPE(COORDINATE_SYSTEM_-
TYPE),pointer *COORDINATE_SYSTEM*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 165 of file region_routines.f90.

7.136 REGION_ROUTINES::REGION_LABEL_SET Interface Reference

Private Member Functions

- subroutine [REGION_LABEL_SET_NUMBER](#) (USER_NUMBER, LABEL, ERR, ERROR,*)
- subroutine [REGION_LABEL_SET_PTR](#) (REGION, LABEL, ERR, ERROR,*)

7.136.1 Detailed Description

Definition at line 77 of file region_routines.f90.

7.136.2 Member Function Documentation

7.136.2.1 subroutine REGION_ROUTINES::REGION_LABEL_SET::REGION_LABEL_SET_NUMBER (INTEGER(INTG),intent(in) *USER_NUMBER*, CHARACTER(LEN=*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 462 of file region_routines.f90.

7.136.2.2 subroutine REGION_ROUTINES::REGION_LABEL_SET::REGION_LABEL_SET_PTR (TYPE(REGION_TYPE),pointer *REGION*, CHARACTER(LEN=*),intent(in) *LABEL*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 492 of file region_routines.f90.

7.137 SORTING::BUBBLE_SORT Interface Reference

Private Member Functions

- subroutine [BUBBLE_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [BUBBLE_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [BUBBLE_SORT_DP](#) (A, ERR, ERROR,*)

7.137.1 Detailed Description

Definition at line 61 of file sorting.f90.

7.137.2 Member Function Documentation

7.137.2.1 subroutine [SORTING::BUBBLE_SORT::BUBBLE_SORT_DP](#)
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 185 of file sorting.f90.

7.137.2.2 subroutine [SORTING::BUBBLE_SORT::BUBBLE_SORT_INTG](#)
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 96 of file sorting.f90.

7.137.2.3 subroutine [SORTING::BUBBLE_SORT::BUBBLE_SORT_SP](#)
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 140 of file sorting.f90.

7.138 SORTING::HEAP_SORT Interface Reference

Private Member Functions

- subroutine [HEAP_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [HEAP_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [HEAP_SORT_DP](#) (A, ERR, ERROR,*)

7.138.1 Detailed Description

Definition at line 67 of file sorting.f90.

7.138.2 Member Function Documentation

7.138.2.1 subroutine SORTING::HEAP_SORT::HEAP_SORT_DP
(REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 362 of file sorting.f90.

7.138.2.2 subroutine SORTING::HEAP_SORT::HEAP_SORT_INTG
(INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 239 of file sorting.f90.

7.138.2.3 subroutine SORTING::HEAP_SORT::HEAP_SORT_SP
(REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*, *) [private]

Definition at line 300 of file sorting.f90.

7.139 SORTING::SHELL_SORT Interface Reference

Private Member Functions

- subroutine `SHELL_SORT_INTG` (*A, ERR, ERROR,**)
- subroutine `SHELL_SORT_SP` (*A, ERR, ERROR,**)
- subroutine `SHELL_SORT_DP` (*A, ERR, ERROR,**)

7.139.1 Detailed Description

Definition at line 73 of file sorting.f90.

7.139.2 Member Function Documentation

7.139.2.1 subroutine `SORTING::SHELL_SORT::SHELL_SORT_DP`
(`REAL(DP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, */` [private])

Definition at line 524 of file sorting.f90.

7.139.2.2 subroutine `SORTING::SHELL_SORT::SHELL_SORT_INTG`
(`INTEGER(INTG),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, */` [private])

Definition at line 433 of file sorting.f90.

7.139.2.3 subroutine `SORTING::SHELL_SORT::SHELL_SORT_SP`
(`REAL(SP),dimension(:),intent(inout) A, INTEGER(INTG),intent(out) ERR,`
`TYPE(VARYING_STRING),intent(out) ERROR, */` [private])

Definition at line 478 of file sorting.f90.

7.140 STRINGS::CHARACTER_TO_LOWERCASE Interface Reference

Returns a character string which is the lowercase equivalent of the supplied string.

Private Member Functions

- function [CHARACTER_TO_LOWERCASE_C](#) (STRING)
- function [CHARACTER_TO_LOWERCASE_VS](#) (STRING)

7.140.1 Detailed Description

Returns a character string which is the lowercase equivalent of the supplied string.

Definition at line 126 of file strings.f90.

7.140.2 Member Function Documentation

**7.140.2.1 function STRINGS::CHARACTER_TO_LOWERCASE::CHARACTER_TO_LOWERCASE_C (CHARACTER(LEN=*)*intent(in)* STRING)
[private]**

Parameters:

STRING The string to convert to lowercase

Definition at line 1609 of file strings.f90.

**7.140.2.2 function STRINGS::CHARACTER_TO_LOWERCASE::CHARACTER_TO_LOWERCASE_VS (TYPE(VARYING_STRING)*intent(in)* STRING)
[private]**

Parameters:

STRING The string to convert

Definition at line 1634 of file strings.f90.

7.141 STRINGS::CHARACTER_TO_UPPERCASE Interface Reference

Returns a character string which is the uppercase equivalent of the supplied string.

Private Member Functions

- function [CHARACTER_TO_UPPERCASE_C](#) (STRING)
- function [CHARACTER_TO_UPPERCASE_VS](#) (STRING)

7.141.1 Detailed Description

Returns a character string which is the uppercase equivalent of the supplied string.

Definition at line 138 of file strings.f90.

7.141.2 Member Function Documentation

7.141.2.1 function STRINGS::CHARACTER_TO_UPPERCASE::CHARACTER_TO_UPPERCASE_C (CHARACTER(LEN=*)*intent(in)* STRING)
[private]

Parameters:

STRING The string to convert

Definition at line 1709 of file strings.f90.

7.141.2.2 function STRINGS::CHARACTER_TO_UPPERCASE::CHARACTER_TO_UPPERCASE_VS (TYPE(VARYING_STRING)*intent(in)* STRING)
[private]

Parameters:

STRING The string to convert

Definition at line 1734 of file strings.f90.

7.142 STRINGS::IS_ABBREVIATION Interface Reference

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.

Private Member Functions

- LOGICAL [IS_ABBREVIATION_C_C](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
- LOGICAL [IS_ABBREVIATION_C_VS](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
- LOGICAL [IS_ABBREVIATION_VS_C](#) (SHORT, LONG, MIN_NUM_CHARACTERS)
- LOGICAL [IS_ABBREVIATION_VS_VS](#) (SHORT, LONG, MIN_NUM_CHARACTERS)

7.142.1 Detailed Description

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.

Definition at line 62 of file strings.f90.

7.142.2 Member Function Documentation

7.142.2.1 LOGICAL STRINGS::IS_ABBREVIATION::IS_ABBREVIATION_C_C (CHARACTER(LEN=*),intent(in) SHORT, CHARACTER(LEN=*),intent(in) LONG, INTEGER(INTG),intent(in) MIN_NUM_CHARACTERS) [private]

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 160 of file strings.f90.

7.142.2.2 LOGICAL STRINGS::IS_ABBREVIATION::IS_ABBREVIATION_C_VS (CHARACTER(LEN=*),intent(in) SHORT, TYPE(VARYING_STRING),intent(in) LONG, INTEGER(INTG),intent(in) MIN_NUM_CHARACTERS) [private]

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 192 of file strings.f90.

7.142.2.3 LOGICAL STRINGS::IS_ABBREVIATION::IS_ABBREVIATION_VS_C (TYPE(VARYING_STRING),intent(in) SHORT, CHARACTER(LEN=*),intent(in) LONG, INTEGER(INTG),intent(in) MIN_NUM_CHARACTERS) [private]

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 224 of file strings.f90.

7.142.2.4 LOGICAL_STRINGS::IS_ABBREVIATION::IS_ABBREVIATION_VS_VS
(**TYPE(VARYING_STRING),intent(in)**) **SHORT**, **TYPE(VARYING_STRING),intent(in)**
LONG, **INTEGER(INTG),intent(in)** **MIN_NUM_CHARACTERS** [private]

Parameters:

SHORT The short form of the string

LONG The long form of the string

MIN_NUM_CHARACTERS The minimum number of characters to match

Definition at line 256 of file strings.f90.

7.143 STRINGS::LIST_TO_CHARACTER Interface Reference

Converts a list to its equivalent character string representation.

Private Member Functions

- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_C](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR, LIST_LENGTHS)
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_INTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_LINTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_L](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_SP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [LIST_TO_CHARACTER_DP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

7.143.1 Detailed Description

Converts a list to its equivalent character string representation.

Definition at line 70 of file strings.f90.

7.143.2 Member Function Documentation

7.143.2.1 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER::LIST_TO_CHARACTER_C (INTEGER(INTG),intent(in) NUMBER_IN_LIST, CHARACTER(LEN=*),dimension(number_in_list),intent(in) LIST, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR, INTEGER(INTG),dimension(number_in_list),intent(in),optional LIST_LENGTHS) [private]

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use. Ignored for character lists.

ERR The error code

ERROR The error string

LIST_LENGTHS LIST_LENGTHS(i). Optional, The length of the i'th list item.

Definition at line 387 of file strings.f90.

**7.143.2.2 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_-
CHARACTER::LIST_TO_CHARACTER_DP (INTEGER(INTG),intent(in)
NUMBER_IN_LIST, REAL(DP),dimension(number_in_list),intent(in) LIST,
CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Parameters:

NUMBER_IN_LIST The number of items in the list
LIST LIST(i). The i'th item in the list
FORMAT The format to use for the conversion
ERR The error code
ERROR The error string

Definition at line 668 of file strings.f90.

**7.143.2.3 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER::LIST_-
TO_CHARACTER_INTG (INTEGER(INTG),intent(in) NUMBER_IN_LIST,
INTEGER(INTG),dimension(number_in_list),intent(in) LIST,
CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Parameters:

NUMBER_IN_LIST The number of items in the list
LIST LIST(i). The i'th item in the list
FORMAT The format to use for the conversion
ERR The error code
ERROR The error string

Definition at line 458 of file strings.f90.

**7.143.2.4 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_-
CHARACTER::LIST_TO_CHARACTER_L (INTEGER(INTG),intent(in)
NUMBER_IN_LIST, LOGICAL,dimension(number_in_list),intent(in) LIST,
CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Parameters:

NUMBER_IN_LIST The number of items in the list
LIST LIST(i). The i'th item in the list
FORMAT The format to use. Ignored for logical lists.
ERR The error code
ERROR The error string

Definition at line 564 of file strings.f90.

**7.143.2.5 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER::LIST_TO_CHARACTER_LINTG (INTEGER(INTG),intent(in) *NUMBER_IN_LIST*,
INTEGER(LINTG),dimension(*number_in_list*),intent(in) *LIST*,
CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use for the conversion

ERR The error code

ERROR The error string

Definition at line 511 of file strings.f90.

**7.143.2.6 CHARACTER(LEN=MAXSTRLEN) STRINGS::LIST_TO_CHARACTER::LIST_TO_CHARACTER_SP (INTEGER(INTG),intent(in) *NUMBER_IN_LIST*, REAL(SP),dimension(*number_in_list*),intent(in) *LIST*,
CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*,
TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Parameters:

NUMBER_IN_LIST The number of items in the list

LIST LIST(i). The i'th item in the list

FORMAT The format to use for the conversion

ERR The error code

ERROR The error string

Definition at line 616 of file strings.f90.

7.144 STRINGS::NUMBER_TO_CHARACTER Interface Reference

Converts a number to its equivalent character string representation.

Private Member Functions

- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_INTG](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_SP](#) (NUMBER, FORMAT, ERR, ERROR)
- CHARACTER(LEN=MAXSTRLEN) [NUMBER_TO_CHARACTER_DP](#) (NUMBER, FORMAT, ERR, ERROR)

7.144.1 Detailed Description

Converts a number to its equivalent character string representation.

Definition at line 80 of file strings.f90.

7.144.2 Member Function Documentation

7.144.2.1 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER::NUMBER_TO_CHARACTER_DP (REAL(DP),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 947 of file strings.f90.

7.144.2.2 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_CHARACTER::NUMBER_TO_CHARACTER_INTG (INTEGER(INTG),intent(in) NUMBER, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 780 of file strings.f90.

**7.144.2.3 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_-
CHARACTER::NUMBER_TO_CHARACTER_LINTG (INTEGER(LINTG),intent(in)
NUMBER, CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out)
ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 817 of file strings.f90.

**7.144.2.4 CHARACTER(LEN=MAXSTRLEN) STRINGS::NUMBER_TO_-
CHARACTER::NUMBER_TO_CHARACTER_SP (REAL(SP),intent(in) NUMBER,
CHARACTER(LEN=*),intent(in) FORMAT, INTEGER(INTG),intent(out) ERR,
TYPE(VARYING_STRING),intent(out) ERROR) [private]**

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 854 of file strings.f90.

7.145 STRINGS::NUMBER_TO_VSTRING Interface Reference

Converts a number to its equivalent varying string representation.

Private Member Functions

- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_INTG (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_LINTG (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_SP (NUMBER, FORMAT, ERR, ERROR)
- TYPE(VARYING_STRING) NUMBER_TO_VSTRING_DP (NUMBER, FORMAT, ERR, ERROR)

7.145.1 Detailed Description

Converts a number to its equivalent varying string representation.

Definition at line 88 of file strings.f90.

7.145.2 Member Function Documentation

7.145.2.1 TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING::NUMBER_TO_VSTRING_DP (REAL(DP),intent(in) *NUMBER*, CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1222 of file strings.f90.

7.145.2.2 TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING::NUMBER_TO_VSTRING_INTG (INTEGER(INTG),intent(in) *NUMBER*, CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1039 of file strings.f90.

7.145.2.3 **TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING::NUMBER_TO_VSTRING_LINTG (INTEGER(LINTG),intent(in) *NUMBER*, CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1082 of file strings.f90.

7.145.2.4 **TYPE(VARYING_STRING) STRINGS::NUMBER_TO_VSTRING::NUMBER_TO_VSTRING_SP (REAL(SP),intent(in) *NUMBER*, CHARACTER(LEN=*),intent(in) *FORMAT*, INTEGER(INTG),intent(out) *ERR*, TYPE(VARYING_STRING),intent(out) *ERROR*) [private]**

Parameters:

NUMBER The number to convert

FORMAT The format to use in the conversion

ERR The error code

ERROR The error string

Definition at line 1125 of file strings.f90.

7.146 `STRINGS::STRING_TO_DOUBLE` Interface Reference

Converts a string representation of a number to a double precision number.

Private Member Functions

- `REAL(DP) STRING_TO_DOUBLE_C (STRING, ERR, ERROR)`
- `REAL(DP) STRING_TO_DOUBLE_VS (STRING, ERR, ERROR)`

7.146.1 Detailed Description

Converts a string representation of a number to a double precision number.

Definition at line 96 of file strings.f90.

7.146.2 Member Function Documentation

7.146.2.1 `REAL(DP) STRINGS::STRING_TO_DOUBLE::STRING_TO_DOUBLE_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]`

Parameters:

STRING The string to convert

ERR The error code !<The error code

ERROR The error string !<The error string

Definition at line 1322 of file strings.f90.

7.146.2.2 `REAL(DP) STRINGS::STRING_TO_DOUBLE::STRING_TO_DOUBLE_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]`

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1349 of file strings.f90.

7.147 STRINGS::STRING_TO_INTEGER Interface Reference

Converts a string representation of a number to an integer.

Private Member Functions

- INTEGER(INTG) [STRING_TO_INTEGER_C](#) (STRING, ERR, ERROR)
- INTEGER(INTG) [STRING_TO_INTEGER_VS](#) (STRING, ERR, ERROR)

7.147.1 Detailed Description

Converts a string representation of a number to an integer.

Definition at line 102 of file strings.f90.

7.147.2 Member Function Documentation

7.147.2.1 INTEGER(INTG) STRINGS::STRING_TO_INTEGER::STRING_TO_INTEGER_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1380 of file strings.f90.

7.147.2.2 INTEGER(INTG) STRINGS::STRING_TO_INTEGER::STRING_TO_INTEGER_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1407 of file strings.f90.

7.148 `STRINGS::STRING_TO_LOGICAL` Interface Reference

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.

Private Member Functions

- LOGICAL `STRING_TO_LOGICAL_C` (`STRING`, `ERR`, `ERROR`)
- LOGICAL `STRING_TO_LOGICAL_VS` (`STRING`, `ERR`, `ERROR`)

7.148.1 Detailed Description

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.

Definition at line 114 of file strings.f90.

7.148.2 Member Function Documentation

7.148.2.1 LOGICAL `STRINGS::STRING_TO_LOGICAL::STRING_TO_LOGICAL_C` (`CHARACTER(LEN=*)`,`intent(in)` `STRING`, `INTEGER(INTG)`,`intent(out)` `ERR`, `TYPE(VARYING_STRING)`,`intent(out)` `ERROR`) [private]

Parameters:

`STRING` The string to convert

`ERR` The error code

`ERROR` The error string

Definition at line 1496 of file strings.f90.

7.148.2.2 LOGICAL `STRINGS::STRING_TO_LOGICAL::STRING_TO_LOGICAL_VS` (`TYPE(VARYING_STRING)`,`intent(in)` `STRING`, `INTEGER(INTG)`,`intent(out)` `ERR`, `TYPE(VARYING_STRING)`,`intent(out)` `ERROR`) [private]

Parameters:

`STRING` The string to convert

`ERR` The error code

`ERROR` The error string

Definition at line 1523 of file strings.f90.

7.149 STRINGS::STRING_TO_LONG_INTEGER Interface Reference

Converts a string representation of a number to a long integer.

Private Member Functions

- INTEGER(LINTG) [STRING_TO_LONG_INTEGER_C](#) (STRING, ERR, ERROR)
- INTEGER(LINTG) [STRING_TO_LONG_INTEGER_VS](#) (STRING, ERR, ERROR)

7.149.1 Detailed Description

Converts a string representation of a number to a long integer.

Definition at line 108 of file strings.f90.

7.149.2 Member Function Documentation

**7.149.2.1 INTEGER(LINTG) STRINGS::STRING_TO_LONG_INTEGER::STRING_TO_LONG_INTEGER_C (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1438 of file strings.f90.

**7.149.2.2 INTEGER(LINTG) STRINGS::STRING_TO_LONG_INTEGER::STRING_TO_LONG_INTEGER_VS (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR)
[private]**

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1465 of file strings.f90.

7.150 **STRINGS::STRING_TO_SINGLE** Interface Reference

Converts a string representation of a number to a single precision number.

Private Member Functions

- REAL(SP) **STRING_TO_SINGLE_C** (STRING, ERR, ERROR)
- REAL(SP) **STRING_TO_SINGLE_VS** (STRING, ERR, ERROR)

7.150.1 Detailed Description

Converts a string representation of a number to a single precision number.

Definition at line 120 of file strings.f90.

7.150.2 Member Function Documentation

7.150.2.1 **REAL(SP) STRINGS::STRING_TO_SINGLE::STRING_TO_SINGLE_C** (CHARACTER(LEN=*),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1552 of file strings.f90.

7.150.2.2 **REAL(SP) STRINGS::STRING_TO_SINGLE::STRING_TO_SINGLE_VS** (TYPE(VARYING_STRING),intent(in) STRING, INTEGER(INTG),intent(out) ERR, TYPE(VARYING_STRING),intent(out) ERROR) [private]

Parameters:

STRING The string to convert

ERR The error code

ERROR The error string

Definition at line 1579 of file strings.f90.

7.151 STRINGS::VSTRING_TO_LOWERCASE Interface Reference

Returns a varying string which is the lowercase equivalent of the supplied string.

Private Member Functions

- TYPE(VARYING_STRING) VSTRING_TO_LOWERCASE_C (STRING)
- TYPE(VARYING_STRING) VSTRING_TO_LOWERCASE_VS (STRING)

7.151.1 Detailed Description

Returns a varying string which is the lowercase equivalent of the supplied string.

Definition at line 132 of file strings.f90.

7.151.2 Member Function Documentation

7.151.2.1 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_LOWERCASE::VSTRING_TO_LOWERCASE_C (CHARACTER(LEN=*)*,intent(in)* STRING)
[private]

Parameters:

STRING The string to convert

Definition at line 1659 of file strings.f90.

7.151.2.2 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_LOWERCASE::VSTRING_TO_LOWERCASE_VS (TYPE(VARYING_STRING)*,intent(in)* STRING)
[private]

Parameters:

STRING The string to convert

Definition at line 1684 of file strings.f90.

7.152 **STRINGS::VSTRING_TO_UPPERCASE** Interface Reference

Returns a varying string which is the uppercase equivalent of the supplied string.

Private Member Functions

- TYPE(VARYING_STRING) VSTRING_TO_UPPERCASE_C (STRING)
- TYPE(VARYING_STRING) VSTRING_TO_UPPERCASE_VS (STRING)

7.152.1 Detailed Description

Returns a varying string which is the uppercase equivalent of the supplied string.

Definition at line 144 of file strings.f90.

7.152.2 Member Function Documentation

7.152.2.1 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_UPPERCASE::VSTRING_TO_UPPERCASE_C (CHARACTER(LEN=*),intent(in) STRING) [private]

Parameters:

STRING The string to convert

Definition at line 1759 of file strings.f90.

7.152.2.2 TYPE(VARYING_STRING) STRINGS::VSTRING_TO_UPPERCASE::VSTRING_TO_UPPERCASE_VS (TYPE(VARYING_STRING),intent(in) STRING) [private]

Parameters:

STRING The string to convert

Definition at line 1784 of file strings.f90.

7.153 TIMER::interface Interface Reference

Private Member Functions

- subroutine [CPUTIMER](#) (RETURN_TIME, TIME_TYPE, ERR, CERROR)

7.153.1 Detailed Description

Definition at line 69 of file timer_f.f90.

7.153.2 Member Function Documentation

**7.153.2.1 subroutine TIMER::interface::CPUTIMER (REAL(DP),intent(out) *RETURN_TIME*,
INTEGER(INTG),intent(in) *TIME_TYPE*, INTEGER(INTG),intent(out) *ERR*,
INTEGER(INTG),dimension(*),intent(out) *CERROR*) [private]**

Definition at line 71 of file timer_f.f90.

References TIMER::CPU_TIMER().

Here is the call graph for this function:

7.154 TREES::TREE_NODE_TYPE Struct Reference

Collaboration diagram for TREES::TREE_NODE_TYPE:

Private Attributes

- INTEGER(INTG) **KEY**
The key for the tree node.
- INTEGER(INTG) **VALUE**
The value stored at the tree node.
- INTEGER(INTG) **COLOUR**
The colour of the node for the red-black tree.
- TYPE(TREE_NODE_TYPE), pointer **LEFT**
The pointer to the left daughter tree node if any.
- TYPE(TREE_NODE_TYPE), pointer **RIGHT**
The pointer to the right daughter tree node if any.
- TYPE(TREE_NODE_TYPE), pointer **PARENT**
The pointer to the parent tree node.

7.154.1 Detailed Description

Definition at line 84 of file trees.f90.

7.154.2 Member Data Documentation

7.154.2.1 INTEGER(INTG) TREES::TREE_NODE_TYPE::COLOUR [private]

The colour of the node for the red-black tree.

Definition at line 88 of file trees.f90.

7.154.2.2 INTEGER(INTG) TREES::TREE_NODE_TYPE::KEY [private]

The key for the tree node.

Definition at line 86 of file trees.f90.

7.154.2.3 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_NODE_TYPE::LEFT [private]

The pointer to the left daughter tree node if any.

Definition at line 89 of file trees.f90.

**7.154.2.4 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_NODE_TYPE::PARENT
[private]**

The pointer to the parent tree node.

Definition at line 91 of file trees.f90.

**7.154.2.5 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_NODE_TYPE::RIGHT
[private]**

The pointer to the right daughter tree node if any.

Definition at line 90 of file trees.f90.

7.154.2.6 INTEGER(INTG) TREES::TREE_NODE_TYPE::VALUE [private]

The value stored at the tree node.

Definition at line 87 of file trees.f90.

7.155 TREES::TREE_TYPE Struct Reference

Collaboration diagram for TREES::TREE_TYPE:

Private Attributes

- LOGICAL TREE_FINISHED
Is .TRUE. if the tree has finished being created, .FALSE. if not.
- INTEGER(INTG) INSERT_TYPE
The insert type for duplicate keys for the tree.
- INTEGER(INTG) NUMBER_IN_TREE
The number of items currently in the tree.
- TYPE(TREE_NODE_TYPE), pointer ROOT
The pointer to the root of the tree.
- TYPE(TREE_NODE_TYPE), pointer NIL
The pointer to the nil of the tree.

7.155.1 Detailed Description

Definition at line 94 of file trees.f90.

7.155.2 Member Data Documentation

7.155.2.1 INTEGER(INTG) TREES::TREE_TYPE::INSERT_TYPE [private]

The insert type for duplicate keys for the tree.

Definition at line 97 of file trees.f90.

7.155.2.2 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_TYPE::NIL [private]

The pointer to the nil of the tree.

Definition at line 100 of file trees.f90.

7.155.2.3 INTEGER(INTG) TREES::TREE_TYPE::NUMBER_IN_TREE [private]

The number of items currently in the tree.

Definition at line 98 of file trees.f90.

7.155.2.4 TYPE(TREE_NODE_TYPE),pointer TREES::TREE_TYPE::ROOT [private]

The pointer to the root of the tree.

Definition at line 99 of file trees.f90.

7.155.2.5 LOGICAL TREES::TREE_TYPE::TREE_FINISHED [private]

Is .TRUE. if the tree has finished being created, .FALSE. if not.

Definition at line 96 of file trees.f90.

7.156 TYPES::BASIS_FUNCTIONS_TYPE Struct Reference

Contains information on the defined basis functions.

Collaboration diagram for TYPES::BASIS_FUNCTIONS_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_BASIS_FUNCTIONS](#)
The number of basis functions definegd.
- TYPE([BASIS_PTR_TYPE](#)), pointer [BASES](#)
The array of pointers to the defined basis functions.

7.156.1 Detailed Description

Contains information on the defined basis functions.

Definition at line 173 of file types.f90.

7.156.2 Member Data Documentation

7.156.2.1 TYPE(BASIS_PTR_TYPE),pointer TYPES::BASIS_FUNCTIONS_TYPE::BASES

The array of pointers to the defined basis functions.

Definition at line 175 of file types.f90.

7.156.2.2 INTEGER(INTG) TYPES::BASIS_FUNCTIONS_TYPE::NUMBER_BASIS_FUNCTIONS

The number of basis functions definegd.

Definition at line 174 of file types.f90.

7.157 TYPES::BASIS_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a BASIS_TYPE.

Collaboration diagram for TYPES::BASIS_PTR_TYPE:

Public Attributes

- TYPE(BASIS_TYPE), pointer PTR

The pointer to the basis.

7.157.1 Detailed Description

A buffer type to allow for an array of pointers to a BASIS_TYPE.

Definition at line 118 of file types.f90.

7.157.2 Member Data Documentation

7.157.2.1 TYPE(BASIS_TYPE),pointer TYPES::BASIS_PTR_TYPE::PTR

The pointer to the basis.

Definition at line 119 of file types.f90.

7.158 TYPES::BASIS_TYPE Struct Reference

Contains all information about a basis .

Collaboration diagram for TYPES::BASIS_TYPE:

Public Attributes

- INTEGER(INTG) [USER_NUMBER](#)

The user defined identifier for the basis. The user number must be unique.

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The global number for the basis i.e., the position identifier for the list of bases defined.

- INTEGER(INTG) [FAMILY_NUMBER](#)

The family number for the basis. A basis has a number of sub-bases attached which make a basis family. The main parent basis is the basis defined by the user and it will have a family number of 0. The sub-bases of the parent basis will be the line and face bases that make up the basis. These will have different family numbers.

- LOGICAL [BASIS_FINISHED](#)

Is .TRUE. if the basis has finished being created, .FALSE. if not.

- LOGICAL [HERMITE](#)

Is .TRUE. if the basis is a hermite basis, .FALSE. if not.

- INTEGER(INTG) [TYPE](#)

The type of basis.

- INTEGER(INTG) [NUMBER_OF_XI](#)

The number of xi directions for the basis.

- INTEGER(INTG) [NUMBER_OF_XI_COORDINATES](#)

The number of xi coordinate directions for the basis. For Lagrange Hermite tensor product basis functions this is equal to the number of Xi directions. For simplex basis functions this is equal to the number of Xi directions + 1.

- INTEGER(INTG), allocatable [INTERPOLATION_XI](#)

INTERPOLATION_XI(ni). The interpolation specification used in the ni'th Xi direction.

- INTEGER(INTG), allocatable [INTERPOLATION_TYPE](#)

INTERPOLATION_TYPE(ni). The interpolation type in the ni'Xi coordinate direction. Old CMISS name IBT(1,ni,nb).

- INTEGER(INTG), allocatable [INTERPOLATION_ORDER](#)

INTERPOLATION_ORDER(ni). The interpolation order in the ni'Xi coordinate direction. Old CMISS name IBT(2,ni,nb).

- LOGICAL [DEGENERATE](#)

Is .TRUE. if the basis is a degenerate basis (i.e., has collapsed nodes), .FALSE. if not.

- INTEGER(INTG), allocatable **COLLAPSED_XI**

COLLAPSED_XI(ni). The collapsed state of the ni'th direction. *COLLAPSED_XI* can be either *XI_COLLAPSED*, *COLLAPSED_AT_XI0*, *COLLAPSED_AT_XII* or *NOT_COLLAPSED* depending on whether or not the ni'th direction is collapsed, has a perpendicular Xi collapsed at the xi=0 end of the ni'th direction, has a perpendicular xi collapsed at the xi=1 of the ni'th direction or is not collapsed. NOTE: in old cmiss the value *IBT(1,ni)* = 5 or 6 was set for the ni that was collapsed. The perpendicular line/face was then stored in *IBT(3,ni)*. For this the quadratic1 and quadratic2 type interpolation types are set on the perpendicular xi direction and the ni direction that is collapsed will have *COLLAPSED_XI(ni)* set to *XI_COLLAPSED*. BE CAREFUL WITH THIS WHEN TRANSLATING OLD CMISS CODE. Old CMISS name *IBT(1,ni) ????*

- INTEGER(INTG) **NUMBER_OF_COLLAPSED_XI**

The number of xi directions in the basis that are collapsed.

- LOGICAL, allocatable **NODE_AT_COLLAPSE**

NODE_AT_COLLAPSE(nn). Is .TRUE. if the nn'th node of the basis is at a collapse, .FALSE. if not.

- TYPE(QUADRATURE_TYPE) **QUADRATURE**

The quadrature schemes for the basis.

- INTEGER(INTG) **NUMBER_OF_PARTIAL_DERIVATIVES**

The number of paratial derivatives for the basis. Old CMISS name NUT(nb).

- INTEGER(INTG) **NUMBER_OF_NODES**

The number of local nodes in the basis. Old CMISS name NNT(nb).

- INTEGER(INTG), allocatable **NUMBER_OF_NODES_XI**

NUMBER_OF_NODES_XI(ni). The number of local nodes in the ni'th direction in the basis. Old CMISS name *IBT(2,ni,nb)*.

- INTEGER(INTG) **NUMBER_OF_ELEMENT_PARAMETERS**

The number of element parameters in the basis. Old CMISS name NST(nb).

- INTEGER(INTG) **MAXIMUM_NUMBER_OF_DERIVATIVES**

The maximum number of derivatives at any node in the basis. Old CMISS name NKT(0,nb).

- INTEGER(INTG), allocatable **NUMBER_OF_DERIVATIVES**

NUMBER_OF_DERIVATIVES(nn). The number of derivatives at the nn'th node in the basis. Old CMISS name *NKT(nn,nb)*.

- INTEGER(INTG), allocatable **NODE_POSITION_INDEX**

NODE_POSITION_INDEX(nn,nic). The index of the node position for the nn'th local node in the nic'th coordinate. For Lagrange-Hermite tensor product basis functions: The number of coordinates equals the number of xi directions. Thus if *NODE_POSITION_INDEX(nn,:)=1,2,2* then local node nn is the first node in the ni(c)=1 direction, the second node in the ni(c)=2 direction and the second node in the ni(c)=3 direction; For simplex basis functions: The number of coordinates equals the number of xi directions plus one. The index specifies the inverse distance away from the corner/end of that area coordinate. Thus if an element has quadratic interpolation the index will range from 3 (closest to the corner/end of the element that the area coordinate has the value 1) to 1 (closest to the corners/end of the element that the area coordinate has the value 0). If M is the order of the element then in *NODE_POSITION_INDEX(nn,:)=1,1,M* then that node is apex for the third area coordinate. In general the index values will add up to M+number of xi directions+1 (i.e., subtract one from the indicies to get the standard simplex coordinates. Old CMISS name *INP(nn,ni,nb)*.

Generated on Wed Oct 22 16:05:26 2008 for openCMISS by Doxygen

- INTEGER(INTG), allocatable [NODE_POSITION_INDEX_INV](#)

NODE_POSITION_INDEX_INV(nnc1,nnc2,nnc3,nnc4). The inverse of the node position index for the basis. The NODE_POSITION_INDEX_INV gives the local node number for the node that has node position indices of nnc1 in the 1st ni(c) direction, nnc2 in the 2nd ni(c) direction, nnc3 in the 3rd ni(c) direction and nnc4 in the 4th ni(c) direction. NOTE: that if the basis has less than 4 ni(c) direction the position index is 1. Old CMISS name NNB(inp1,inp2,inp3,nbf).

- INTEGER(INTG), allocatable [DERIVATIVE_ORDER_INDEX](#)

DERIVATIVE_ORDER_INDEX(nk,nn,0:ni,nbf). The index of the derivative order for the nk'th derivative of the nn'th node in the ni'th direction of the basis. The derivative index is NO_PART_DERIV for zeroth order, FIRST_PART_DERIV for the first order and SECOND_PART_DERIV for the second order derivative. Thus a DERIVATIVE_ORDER_INDEX(nk,nn,1..) of {NO_PART_DERIV,FIRST_PART_DERIV,NO_PART_DERIV} indicates that the nk'th derivative of the nn'th node of the basis is the first derivative with respect to the s2 direction. Old CMISS name IDO(nk,nn,1:ni,nbf).

- INTEGER(INTG), allocatable [DERIVATIVE_ORDER_INDEX_INV](#)

DERIVATIVE_ORDER_INDEX_INV(nu1,nu2,nu3,nn). The inverse of the derivative order index for the nn'th local node of the basis. DERIVATIVE_ORDER_INDEX_INV gives the derivative number for the nu1 partial derivative in the 1st xi direction, the nu2 partial derivative in the 2nd xi direction and the nu3 partial derivative in the 3rd xi direction. NOTE: local node nn does not carry any derivatives of the requested partial derivative type then DERIVATIVE_ORDER_INDEX_INV will return 0. If the basis has less than 3 xi directions then the nu index is 1.

- INTEGER(INTG), allocatable [PARTIAL_DERIVATIVE_INDEX](#)

PARTIAL_DERIVATIVE_INDEX(nk,nn). Gives the partial derivative number (nu) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name IDO(nk,nn,0,nbf).

- INTEGER(INTG), allocatable [ELEMENT_PARAMETER_INDEX](#)

ELEMENT_PARAMETER_INDEX(nk,nn). Gives the element parameter number (ns) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name NSB(nk,nn,nbf).

- INTEGER(INTG) [NUMBER_OF_LOCAL_LINES](#)

The number of local lines in the basis.

- INTEGER(INTG), allocatable [LOCAL_LINE_XI_DIRECTION](#)

LOCAL_LINE_XI_DIRECTION(nae). The Xi direction of the nae'th local line for the basis.

- INTEGER(INTG), allocatable [NUMBER_OF_NODES_IN_LOCAL_LINE](#)

NUMBER_OF_NODES_IN_LOCAL_LINE(nae). The the number of nodes in the nae'th local line for the basis. Old CMISS name NNL(0,nae,nb).

- INTEGER(INTG), allocatable [NODE_NUMBERS_IN_LOCAL_LINE](#)

NODE_NUMBERS_IN_LOCAL_LINE(nnl,nae). The local node numbers (nn) for the nnl'th line node in the nae'th local line for the basis. Old CMISS name NNL(1..,nae,nb).

- INTEGER(INTG), allocatable [DERIVATIVE_NUMBERS_IN_LOCAL_LINE](#)

DERIVATIVES_NUMBERS_IN_LOCAL_LINE(nnl,nae). The derivative numbers (nk) for the nnl'th line node in the nae'th local line for the basis.

- [TYPE\(BASIS_PTR_TYPE\)](#), pointer [LINE_BASES](#)

LINE_BASES(nae). The pointer to the basis for the nae'th line for the basis.

- [TYPE\(BASIS_PTR_TYPE\)](#), pointer [FACE_BASES](#)

FACE_BASES(naf). The pointer to the basis for the naf'th face for the basis.

- [INTEGER\(INTG\) NUMBER_OF_SUB_BASES](#)

The number of sub-bases (lines, faces) for the basis.

- [TYPE\(BASIS_PTR_TYPE\)](#), pointer [SUB_BASES](#)

SUB_BASES(sbn). The pointer to the sbn'th sub-basis for the basis.

- [TYPE\(BASIS_TYPE\)](#), pointer [PARENT_BASIS](#)

The pointer to the parent basis for the basis. NOTE: that if the basis is not a sub-basis of another basis this pointer will be NULL.

7.158.1 Detailed Description

Contains all information about a basis .

Definition at line 123 of file types.f90.

7.158.2 Member Data Documentation

7.158.2.1 LOGICAL TYPES::BASIS_TYPE::BASIS_FINISHED

Is .TRUE. if the basis has finished being created, .FALSE. if not.

Definition at line 128 of file types.f90.

7.158.2.2 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::COLLAPSED_XI

COLLAPSED_XI(ni). The collapsed state of the ni'th direction. COLLAPSED_XI can be either XI_COLLAPSED, COLLAPSED_AT_XI0, COLLAPSED_AT_XI1 or NOT_COLLAPSED depending on whether or not the ni'th direction is collapsed, has a perpendicular Xi collapsed at the xi=0 end of the ni'th direction, has a perpendicular xi collapsed at the xi=1 of the ni'th direction or is not collapsed. NOTE: in old cmiss the value IBT(1,ni) = 5 or 6 was set for the ni that was collapsed. The perpendicular line/face was then stored in IBT(3,ni). For this the quadratic1 and quadratic2 type interpolation types are set on the perpendicular xi direction and the ni direction that is collapsed will have COLLAPSED_XI(ni) set to XI_COLLAPSED. BE CAREFUL WITH THIS WHEN TRANSLATING OLD CMISS CODE. Old CMISS name IBT(1,ni) ????

See also:

[BASIS_ROUTINES_XiCollapse](#)

Definition at line 138 of file types.f90.

7.158.2.3 LOGICAL TYPES::BASIS_TYPE::DEGENERATE

Is .TRUE. if the basis is a degenerate basis (i.e., has collapsed nodes), .FALSE. if not.

Definition at line 137 of file types.f90.

7.158.2.4 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::DERIVATIVE_NUMBERS_IN_LOCAL_LINE

DERIVATIVES_NUMBERS_IN_LOCAL_LINE(nnl,nae). The derivative numbers (nk) for the nnl'th line node in the nae'th local line for the basis.

Definition at line 161 of file types.f90.

7.158.2.5 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::DERIVATIVE_ORDER_INDEX

DERIVATIVE_ORDER_INDEX(nk,nn,0:ni,nbf). The index of the derivative order for the nk'th derivative of the nn'th node in the ni'th direction of the basis. The derivative index is NO_PART_DERIV for zeroth order, FIRST_PART_DERIV for the first order and SECOND_PART_DERIV for the second order derivative. Thus a DERIVATIVE_ORDER_INDEX(nk,nn,1..) of {NO_PART_DERIV,FIRST_PART_DERIV,NO_PART_DERIV} indicates that the nk'th derivative of the nn'th node of the basis is the first derivative with respect to the s2 direction. Old CMISS name IDO(nk,nn,1:ni,nbf).

See also:

[TYPES::BASIS_TYPE::DERIVATIVE_ORDER_INDEX_INV](#),
CONSTANTS_PartialDerivativeConstants

Definition at line 152 of file types.f90.

7.158.2.6 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::DERIVATIVE_ORDER_INDEX_INV

DERIVATIVE_ORDER_INDEX_INV(nu1,nu2,nu3,nn). The inverse of the derivative order index for the nn'th local node of the basis. DERIVATIVE_ORDER_INDEX_INV gives the derivative number for the nu1 partial derivative in the 1st xi direction, the nu2 partial derivative in the 2nd xi direction and the nu3 partial derivative in the 3rd xi direction. NOTE: local node nn does not carry any derivatives of the requested partial derivative type then DERIVATIVE_ORDER_INDEX_INV will return 0. If the basis has less than 3 xi directions then the nu index is 1.

See also:

[TYPES::BASIS_TYPE::DERIVATIVE_ORDER_INDEX](#)

Definition at line 153 of file types.f90.

7.158.2.7 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::ELEMENT_PARAMETER_INDEX

ELEMENT_PARAMETER_INDEX(nk,nn). Gives the element parameter number (ns) of the nk'th derivative of the nn'th local node for the basis. Old CMISS name NSB(nk,nn,nbf).

Definition at line 155 of file types.f90.

7.158.2.8 TYPE(BASIS_PTR_TYPE),pointer TYPES::BASIS_TYPE::FACE_BASES

FACE_BASES(naf). The pointer to the basis for the naf'th face for the basis.

Definition at line 166 of file types.f90.

7.158.2.9 INTEGER(INTG) TYPES::BASIS_TYPE::FAMILY_NUMBER

The family number for the basis. A basis has a number of sub-bases attached which make a basis family. The main parent basis is the basis defined by the user and it will have a family number of 0. The sub-bases of the parent basis will be the line and face bases that make up the basis. These will have different family numbers.

Definition at line 127 of file types.f90.

7.158.2.10 INTEGER(INTG) TYPES::BASIS_TYPE::GLOBAL_NUMBER

The global number for the basis i.e., the position identifier for the list of bases defined.

Definition at line 126 of file types.f90.

7.158.2.11 LOGICAL TYPES::BASIS_TYPE::HERMITE

Is .TRUE. if the basis is a hermite basis, .FALSE. if not.

Definition at line 129 of file types.f90.

7.158.2.12 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::INTERPOLATION_ORDER

INTERPOLATION_ORDER(ni). The interpolation order in the ni'th Xi coordinate direction. Old [CMISS](#) name IBT(2,ni,nb).

See also:

[BASIS_ROUTINES_InterpolationOrder](#)

Definition at line 135 of file types.f90.

7.158.2.13 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::INTERPOLATION_TYPE

INTERPOLATION_TYPE(ni). The interpolation type in the ni'th Xi coordinate direction. Old [CMISS](#) name IBT(1,ni,nb).

See also:

[BASIS_ROUTINES_InterpolationTypes](#)

Definition at line 134 of file types.f90.

7.158.2.14 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::INTERPOLATION_XI

INTERPOLATION_XI(ni). The interpolation specification used in the ni'th Xi direction.

See also:

[BASIS_ROUTINES_InterpolationSpecifications](#)

Definition at line 133 of file types.f90.

7.158.2.15 TYPE(BASIS_PTR_TYPE),pointer TYPES::BASIS_TYPE::LINE_BASES

LINE_BASES(nae). The pointer to the basis for the nae'th line for the basis.

Definition at line 165 of file types.f90.

7.158.2.16 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::LOCAL_LINE_XI - DIRECTION

LOCAL_LINE_XI_DIRECTION(nae). The Xi direction of the nae'th local line for the basis.

Definition at line 158 of file types.f90.

7.158.2.17 INTEGER(INTG) TYPES::BASIS_TYPE::MAXIMUM_NUMBER_OF - DERIVATIVES

The maximum number of derivatives at any node in the basis. Old [CMISS](#) name NKT(0,nbf).

Definition at line 148 of file types.f90.

7.158.2.18 LOGICAL,allocatable TYPES::BASIS_TYPE::NODE_AT_COLLAPSE

NODE_AT_COLLAPSE(nn). Is .TRUE. if the nn'th node of the basis is at a collapse, .FALSE. if not.

Definition at line 140 of file types.f90.

7.158.2.19 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NODE_NUMBERS_IN - LOCAL_LINE

NODE_NUMBERS_IN_LOCAL_LINE(nnl,nae). The local node numbers (nn) for the nnl'th line node in the nae'th local line for the basis. Old [CMISS](#) name NNL(1..,nae,nb).

Definition at line 160 of file types.f90.

7.158.2.20 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NODE_POSITION_INDEX

NODE_POSITION_INDEX(nn,ni). The index of the node position for the nn'th local node in the ni'th coordinate. For Lagrange-Hermite tensor product basis functions: The number of coordinates equals the number of xi directions. Thus if NODE_POSITION_INDEX(nn,:)=1,2,2 then local node nn is the first node in the ni(c)=1 direction, the second node in the ni(c)=2 direction and the second node in the ni(c)=3 direction; For simplex basis functions: The number of coordinates equals the number of xi directions plus one. The index specifies the inverse distance away from the corner/end of that area coordinate. Thus if an element has quadratic interpolation the index will range from 3 (closest to the corner/end of the element that the area coordinate has the value 1) to 1 (closest to the corners/end of the element that the area coordinate has the value 0). If M is the order of the element then in NODE_POSITION_INDEX(nn,:)=1,1,M then that node is apex for the third area coordinate. In general the index values will add up to M+number of xi directions+1 (i.e., subtract one from the indices to get the standard simplex coordinates. Old [CMISS](#) name INP(nn,ni,nb).

See also:

[TYPES::BASIS_TYPE::NODE_POSITION_INDEX_INV](#).

Definition at line 150 of file types.f90.

7.158.2.21 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NODE_POSITION_INDEX_INV

NODE_POSITION_INDEX_INV(nnc1,nnc2,nnc3,nnc4). The inverse of the node position index for the basis. The NODE_POSITION_INDEX_INV gives the local node number for the node that has node position indices of nnc1 in the 1st ni(c) direction, nnc2 in the 2nd ni(c) direction, nnc3 in the 3rd ni(c) direction and nnc4 in the 4th ni(c) direction. NOTE: that if the basis has less than 4 ni(c) direction the position index is 1. Old [CMISS](#) name NNB(inp1,inp2,inp3,nbf).

See also:

[TYPES::BASIS_TYPE::NODE_POSITION_INDEX](#).

Definition at line 151 of file types.f90.

7.158.2.22 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_COLLAPSED_XI

The number of xi directions in the basis that are collapsed.

Definition at line 139 of file types.f90.

7.158.2.23 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NUMBER_OF_DERIVATIVES

NUMBER_OF_DERIVATIVES(nn). The number of derivatives at the nn'th node in the basis. Old [CMISS](#) name NKT(nn,nbf).

Definition at line 149 of file types.f90.

7.158.2.24 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_ELEMENT_PARAMETERS

The number of element parameters in the basis. Old [CMISS](#) name NST(nbf).

Definition at line 147 of file types.f90.

7.158.2.25 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_LOCAL_LINES

The number of local lines in the basis.

Definition at line 157 of file types.f90.

7.158.2.26 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_NODES

The number of local nodes in the basis. Old [CMISS](#) name NNT(nbf).

Definition at line 144 of file types.f90.

7.158.2.27 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NUMBER_OF_NODES_IN_LOCAL_LINE

NUMBER_OF_NODES_IN_LOCAL_LINE(nae). The the number of nodes in the nae'th local line for the basis. Old [CMISS](#) name NNL(0,nae,nb).

Definition at line 159 of file types.f90.

7.158.2.28 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::NUMBER_OF_NODES_XI

NUMBER_OF_NODES_XI(ni). The number of local nodes in the ni'th direction in the basis. Old [CMISS](#) name IBT(2,ni,nb).

[Todo](#)

CHANGE TO NUMBER_OF_NODES_XIC i.e., xi coordinate index not xi???

Definition at line 146 of file types.f90.

7.158.2.29 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_PARTIAL_DERIVATIVES

The number of paratial derivatives for the basis. Old [CMISS](#) name NUT(nbf).

Definition at line 143 of file types.f90.

7.158.2.30 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_SUB_BASES

The number of sub-bases (lines, faces) for the basis.

Definition at line 167 of file types.f90.

7.158.2.31 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_XI

The number of xi directions for the basis.

Definition at line 131 of file types.f90.

7.158.2.32 INTEGER(INTG) TYPES::BASIS_TYPE::NUMBER_OF_XI_COORDINATES

The number of xi coordinate directions for the basis. For Lagrange Hermite tensor product basis functions this is equal to the number of Xi directions. For simplex basis functions this is equal to the number of Xi directions + 1.

Definition at line 132 of file types.f90.

7.158.2.33 TYPE(BASIS_TYPE),pointer TYPES::BASIS_TYPE::PARENT_BASIS

The pointer to the parent basis for the basis. NOTE: that if the basis is not a sub-basis of another basis this pointer will be NULL.

Definition at line 169 of file types.f90.

7.158.2.34 INTEGER(INTG),allocatable TYPES::BASIS_TYPE::PARTIAL_DERIVATIVE_INDEX

PARTIAL_DERIVATIVE_INDEX(nk,nn). Gives the partial derivative number (nu) of the nk'th derivative of the nn'th local node for the basis. Old [CMISS](#) name IDO(nk,nn,0,nbf).

Definition at line 154 of file types.f90.

7.158.2.35 TYPE(QUADRATURE_TYPE) TYPES::BASIS_TYPE::QUADRATURE

The quadrature schemes for the basis.

Definition at line 142 of file types.f90.

7.158.2.36 TYPE(BASIS_PTR_TYPE),pointer TYPES::BASIS_TYPE::SUB_BASES

SUB_BASES(sbn). The pointer to the sbn'th sub-basis for the basis.

Definition at line 168 of file types.f90.

7.158.2.37 INTEGER(INTG) TYPES::BASIS_TYPE::TYPE

The type of basis.

See also:

BASIS_ROUTINES_BasisTypes

Definition at line 130 of file types.f90.

7.158.2.38 INTEGER(INTG) TYPES::BASIS_TYPE::USER_NUMBER

The user defined identifier for the basis. The user number must be unique.

Definition at line 125 of file types.f90.

7.159 TYPES::COORDINATE_SYSTEM_TYPE Struct Reference

Contains information on a coordinate system.

Public Attributes

- INTEGER(INTG) [USER_NUMBER](#)

The user defined identifier for the coordinate. The user number must be unique.

- LOGICAL [COORDINATE_SYSTEM_FINISHED](#)

Is .TRUE. if the coordinate system has finished being created, .FALSE. if not.

- INTEGER(INTG) [TYPE](#)

The type of coordinate system. Old [CMISS](#) name ITYP10(nr).

- INTEGER(INTG) [RADIAL_INTERPOLATION_TYPE](#)

The type of radial interpolation type for non-rectangular cartesian systems. Old [CMISS](#) name JTYP10(nr).

- INTEGER(INTG) [NUMBER_OF_DIMENSIONS](#)

The number of dimensions for the coordinate system. Old [CMISS](#) name NJT.

- REAL(DP) [FOCUS](#)

The focus of the coordinate system for a prolate-spheriodal coordinate system.

- REAL(DP) [ORIGIN](#)

*ORIGIN(nj). The nj'th component of the origin of the coordinate system wrt the global coordinate system.
NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.*

- REAL(DP) [ORIENTATION](#)

ORIENTATION(nj,mj). he orientation matrix for the orientation of the coordinate system wrt to the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.

7.159.1 Detailed Description

Contains information on a coordinate system.

Todo

Have a list of coordinate systems and have a pointer in the coordinate_system_type back to the regions that use them.

Definition at line 185 of file types.f90.

7.159.2 Member Data Documentation

7.159.2.1 LOGICAL TYPES::COORDINATE_SYSTEM_TYPE::COORDINATE_SYSTEM_FINISHED

Is .TRUE. if the coordinate system has finished being created, .FALSE. if not.

Definition at line 187 of file types.f90.

7.159.2.2 REAL(DP) TYPES::COORDINATE_SYSTEM_TYPE::FOCUS

The focus of the coordinate system for a prolate-spheriodal coordinate system.

Definition at line 191 of file types.f90.

7.159.2.3 INTEGER(INTG) TYPES::COORDINATE_SYSTEM_TYPE::NUMBER_OF_DIMENSIONS

The number of dimensions for the coordinate system. Old [CMISS](#) name NJT.

Definition at line 190 of file types.f90.

7.159.2.4 REAL(DP) TYPES::COORDINATE_SYSTEM_TYPE::ORIENTATION

ORIENTATION(nj,mj). he orientation matrix for the orientation of the coordinate system wrt to the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.

Definition at line 193 of file types.f90.

7.159.2.5 REAL(DP) TYPES::COORDINATE_SYSTEM_TYPE::ORIGIN

ORIGIN(nj). The nj'th component of the origin of the coordinate system wrt the global coordinate system. NOTE: maybe this should be wrt to the parent regions coordinate system - this would then go into the REGION type.

Definition at line 192 of file types.f90.

7.159.2.6 INTEGER(INTG) TYPES::COORDINATE_SYSTEM_TYPE::RADIAL_INTERPOLATION_TYPE

The type of radial interpolation type for non-rectangular cartesian systems. Old [CMISS](#) name JTYP10(nr).

See also:

[COORDINATE_ROUTINES::RadialInterpolations](#)

Definition at line 189 of file types.f90.

7.159.2.7 INTEGER(INTG) TYPES::COORDINATE_SYSTEM_TYPE::TYPE

The type of coordinate system. Old [CMISS](#) name ITYP10(nr).

See also:

[COORDINATE_ROUTINES::CoordinateSystemTypes](#)

Definition at line 188 of file types.f90.

7.159.2.8 INTEGER(INTG) TYPES::COORDINATE_SYSTEM_TYPE::USER_NUMBER

The user defined identifier for the coordinate. The user number must be unique.

Definition at line 186 of file types.f90.

7.160 TYPES::DECOMPOSITION_ELEMENT_TYPE Struct Reference

Contains the information for an element in a decomposition.

Public Attributes

- INTEGER(INTG) [LOCAL_NUMBER](#)

The local element number in the decomposition.

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The corresponding global element number in the mesh of the local element number in the decomposition.

- INTEGER(INTG) [USER_NUMBER](#)

The corresponding user number for the element.

- INTEGER(INTG), allocatable [NUMBER_OF_ADJACENT_ELEMENTS](#)

NUMBER_OF_ADJACENT_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old CMISS name NXI(-ni:ni,0:nei,ne).

- INTEGER(INTG), allocatable [ADJACENT_ELEMENTS](#)

ADJACENT_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).

- INTEGER(INTG), allocatable [ELEMENT_LINES](#)

ELEMENT_LINES(nae). The local decomposition line number corresponding to the nae'th local line of the element. Old CMISS name NLL(nae,ne).

7.160.1 Detailed Description

Contains the information for an element in a decomposition.

Definition at line 682 of file types.f90.

7.160.2 Member Data Documentation

7.160.2.1 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_ELEMENT_TYPE::ADJACENT_ELEMENTS

ADJACENT_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).

Definition at line 687 of file types.f90.

**7.160.2.2 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_ELEMENT_-
TYPE::ELEMENT_LINES**

ELEMENT_LINES(nae). The local decomposition line number corresponding to the nae'th local line of the element. Old [CMISS](#) name NLL(nae,ne).

Definition at line 688 of file types.f90.

**7.160.2.3 INTEGER(INTG) TYPES::DECOMPOSITION_ELEMENT_TYPE::GLOBAL_-
NUMBER**

The corresponding global element number in the mesh of the local element number in the decomposition.

Definition at line 684 of file types.f90.

**7.160.2.4 INTEGER(INTG) TYPES::DECOMPOSITION_ELEMENT_TYPE::LOCAL_-
NUMBER**

The local element number in the decomposition.

Definition at line 683 of file types.f90.

**7.160.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_ELEMENT_-
TYPE::NUMBER_OF_ADJACENT_ELEMENTS**

NUMBER_OF_ADJACENT_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old [CMISS](#) name NXI(-ni:ni,0:nei,ne).

Definition at line 686 of file types.f90.

7.160.2.6 INTEGER(INTG) TYPES::DECOMPOSITION_ELEMENT_TYPE::USER_NUMBER

The corresponding user number for the element.

Definition at line 685 of file types.f90.

7.161 TYPES::DECOMPOSITION_ELEMENTS_TYPE Struct Reference

Contains the topology information for the elements of a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION_ELEMENTS_TYPE:

Public Attributes

- TYPE(DECOMPOSITION_TYPE), pointer DECOMPOSITION

The pointer to the decomposition for this elements topology information.

- INTEGER(INTG) TOTAL_NUMBER_OF_ELEMENTS

The total number of elements in this decomposition topology.

- TYPE(DECOMPOSITION_ELEMENT_TYPE), pointer ELEMENTS

ELEMENTS(ne). The pointer to the array of topology information for the elements of this decomposition. ELEMENTS(ne) contains the topological information for the ne'th local element of the decomposition.

7.161.1 Detailed Description

Contains the topology information for the elements of a decomposition.

Definition at line 692 of file types.f90.

7.161.2 Member Data Documentation

7.161.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::DECOMPOSITION_ELEMENTS_TYPE::DECOMPOSITION

The pointer to the decomposition for this elements topology information.

Definition at line 693 of file types.f90.

7.161.2.2 TYPE(DECOMPOSITION_ELEMENT_TYPE),pointer TYPES::DECOMPOSITION_ELEMENTS_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of topology information for the elements of this decomposition. ELEMENTS(ne) contains the topological information for the ne'th local element of the decomposition.

Todo

Change this to allocatable???

Definition at line 695 of file types.f90.

**7.161.2.3 INTEGER(INTG) TYPES::DECOMPOSITION_ELEMENTS_TYPE::TOTAL_-
NUMBER_OF_ELEMENTS**

The total number of elements in this decomposition topology.

Definition at line 694 of file types.f90.

7.162 TYPES::DECOMPOSITION_LINE_TYPE Struct Reference

Contains the information for a line in a decomposition.

Public Attributes

- INTEGER(INTG) **NUMBER**
The line number in the decomposition.
- INTEGER(INTG) **XI_DIRECTION**
The Xi direction of the line. Old CMISS name NPL(1,0,nl).
- INTEGER(INTG) **NUMBER_OF_SURROUNDING_ELEMENTS**
The number of elements that surround (use) this line.
- INTEGER(INTG), allocatable **SURROUNDING_ELEMENTS**
SURROUNDING_ELEMENTS(nel). The local element number of the nel'th element that surrounds (uses) this line.
- INTEGER(INTG), allocatable **ELEMENT_LINES**
ELEMENT_LINES(nel). The local arc number of the nel'th element that surrounds (uses) this line.
- INTEGER(INTG) **ADJACENT_LINES**
ADJACENT_LINES(0:1). The line number of adjacent lines. ADJACENT_LINES(0) is the line number adjacent in the -xi direction. ADJACENT_LINES(1) is the line number adjacent in the +xi direction. Old CMISS name NPL(2..3,0,nl).

7.162.1 Detailed Description

Contains the information for a line in a decomposition.

Definition at line 665 of file types.f90.

7.162.2 Member Data Documentation

7.162.2.1 INTEGER(INTG) TYPES::DECOMPOSITION_LINE_TYPE::ADJACENT_LINES

ADJACENT_LINES(0:1). The line number of adjacent lines. ADJACENT_LINES(0) is the line number adjacent in the -xi direction. ADJACENT_LINES(1) is the line number adjacent in the +xi direction. Old CMISS name NPL(2..3,0,nl).

Definition at line 671 of file types.f90.

7.162.2.2 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_LINE_TYPE::ELEMENT_LINES

ELEMENT_LINES(nel). The local arc number of the nel'th element that surrounds (uses) this line.

Definition at line 670 of file types.f90.

7.162.2.3 INTEGER(INTG) TYPES::DECOMPOSITION_LINE_TYPE::NUMBER

The line number in the decomposition.

Definition at line 666 of file types.f90.

**7.162.2.4 INTEGER(INTG) TYPES::DECOMPOSITION_LINE_TYPE::NUMBER_OF_-
SURROUNDING_ELEMENTS**

The number of elements that surround (use) this line.

Definition at line 668 of file types.f90.

**7.162.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_LINE_-
TYPE::SURROUNDING_ELEMENTS**

SURROUNDING_ELEMENTS(nel). The local element number of the nel'th element that surrounds (uses) this line.

Definition at line 669 of file types.f90.

7.162.2.6 INTEGER(INTG) TYPES::DECOMPOSITION_LINE_TYPE::XI_DIRECTION

The Xi direction of the line. Old [CMISS](#) name NPL(1,0,nl).

Definition at line 667 of file types.f90.

7.163 TYPES::DECOMPOSITION_LINES_TYPE Struct Reference

Contains the topology information for the lines of a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION_LINES_TYPE:

Public Attributes

- TYPE(DECOMPOSITION_TYPE), pointer DECOMPOSITION
The pointer to the decomposition for this lines topology information.
- INTEGER(INTG) NUMBER_OF_LINES
The number of lines in this decomposition topology.
- TYPE(DECOMPOSITION_LINE_TYPE), allocatable LINES
LINES(nl). The pointer to the array of topology information for the lines of this decomposition. LINES(nl) contains the topological information for the nl'th local line of the decomposition.

7.163.1 Detailed Description

Contains the topology information for the lines of a decomposition.

Definition at line 675 of file types.f90.

7.163.2 Member Data Documentation

7.163.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::DECOMPOSITION_LINES_TYPE::DECOMPOSITION

The pointer to the decomposition for this lines topology information.

Definition at line 676 of file types.f90.

7.163.2.2 TYPE(DECOMPOSITION_LINE_TYPE),allocatable TYPES::DECOMPOSITION_LINES_TYPE::LINES

LINES(nl). The pointer to the array of topology information for the lines of this decomposition. LINES(nl) contains the topological information for the nl'th local line of the decomposition.

Definition at line 678 of file types.f90.

7.163.2.3 INTEGER(INTG) TYPES::DECOMPOSITION_LINES_TYPE::NUMBER_OF_LINES

The number of lines in this decomposition topology.

Definition at line 677 of file types.f90.

7.164 TYPES::DECOMPOSITION_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DECOMPOSITION_TYPE](#).

Collaboration diagram for TYPES::DECOMPOSITION_PTR_TYPE:

Public Attributes

- TYPE([DECOMPOSITION_TYPE](#)), pointer PTR

The pointer to the domain decomposition.

7.164.1 Detailed Description

A buffer type to allow for an array of pointers to a [DECOMPOSITION_TYPE](#).

Definition at line 722 of file types.f90.

7.164.2 Member Data Documentation

7.164.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::DECOMPOSITION_PTR_- TYPE::PTR

The pointer to the domain decomposition.

Definition at line 723 of file types.f90.

7.165 TYPES::DECOMPOSITION_TOPOLOGY_TYPE Struct Reference

Contains the topology information for a decomposition.

Collaboration diagram for TYPES::DECOMPOSITION_TOPOLOGY_TYPE:

Public Attributes

- TYPE(DECOMPOSITION_TYPE), pointer DECOMPOSITION
The pointer to the decomposition for this topology information.
- TYPE(DECOMPOSITION_ELEMENTS_TYPE), pointer ELEMENTS
The pointer to the topology information for the elements of this decomposition.
- TYPE(DECOMPOSITION_LINES_TYPE), pointer LINES
The pointer to the topology information for the lines of this decomposition.

7.165.1 Detailed Description

Contains the topology information for a decomposition.

Definition at line 699 of file types.f90.

7.165.2 Member Data Documentation

7.165.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::DECOMPOSITION_TOPOLOGY_TYPE::DECOMPOSITION

The pointer to the decomposition for this topology information.

Definition at line 700 of file types.f90.

7.165.2.2 TYPE(DECOMPOSITION_ELEMENTS_TYPE),pointer TYPES::DECOMPOSITION_TOPOLOGY_TYPE::ELEMENTS

The pointer to the topology information for the elements of this decomposition.

Definition at line 701 of file types.f90.

7.165.2.3 TYPE(DECOMPOSITION_LINES_TYPE),pointer TYPES::DECOMPOSITION_TOPOLOGY_TYPE::LINES

The pointer to the topology information for the lines of this decomposition.

Definition at line 702 of file types.f90.

7.166 TYPES::DECOMPOSITION_TYPE Struct Reference

Contains information on the domain decomposition.

Collaboration diagram for TYPES::DECOMPOSITION_TYPE:

Public Attributes

- INTEGER(INTG) [USER_NUMBER](#)
The user defined identifier for the domain decomposition. The user number must be unique.
- INTEGER(INTG) [GLOBAL_NUMBER](#)
The global number of the domain decomposition in the list of domain decompositions for a particular mesh.
- LOGICAL [DECOMPOSITION_FINISHED](#)
Is .TRUE. if the decomposition has finished being created, .FALSE. if not.
- TYPE([DECOMPOSITIONS_TYPE](#)), pointer [DECOMPOSITIONS](#)
A pointer to the decompositions for this decomposition.
- TYPE([MESH_TYPE](#)), pointer [MESH](#)
A pointer to the mesh for this decomposition.
- INTEGER(INTG) [MESH_COMPONENT_NUMBER](#)
The component number (index) of the mesh component that this decomposition belongs to (i.e., was generated from).
- INTEGER(INTG) [DECOMPOSITION_TYPE](#)
The type of the domain decomposition.
- INTEGER(INTG) [NUMBER_OF_DOMAINS](#)
The number of domains that this decomposition contains.
- INTEGER(INTG) [NUMBER_OF_EDGES_CUT](#)
For automatically calculated decompositions, the number of edges of the mesh dual graph that were cut for the composition. It provides an indication of the optimality of the automatic decomposition.
- INTEGER(INTG), allocatable [ELEMENT_DOMAIN](#)
*ELEMENT_DOMAIN(ne). The domain number that the ne'th global element is in for the decomposition.
Note: the domain numbers start at 0 and go up to the NUMBER_OF_DOMAINS-1.*
- TYPE([DECOMPOSITION_TOPOLOGY_TYPE](#)), pointer [TOPOLOGY](#)
A pointer to the topology for this decomposition.
- TYPE([DOMAIN_PTR_TYPE](#)), pointer [DOMAIN](#)
DOMAIN(mesh_component_idx). A pointer to the domain for mesh component for the domain associated with the computational node.

7.166.1 Detailed Description

Contains information on the domain decomposition.

Definition at line 706 of file types.f90.

7.166.2 Member Data Documentation

7.166.2.1 LOGICAL TYPES::DECOMPOSITION_TYPE::DECOMPOSITION_FINISHED

Is .TRUE. if the decomposition has finished being created, .FALSE. if not.

Definition at line 709 of file types.f90.

7.166.2.2 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::DECOMPOSITION_TYPE

The type of the domain decomposition.

See also:

[MESH_ROUTINES::DecompositionTypes](#).

Definition at line 713 of file types.f90.

7.166.2.3 TYPE(DECOMPOSITIONS_TYPE),pointer TYPES::DECOMPOSITION_- TYPE::DECOMPOSITIONS

A pointer to the decompositions for this decomposition.

Definition at line 710 of file types.f90.

7.166.2.4 TYPE(DOMAIN_PTR_TYPE),pointer TYPES::DECOMPOSITION_TYPE::DOMAIN

DOMAIN(mesh_component_idx). A pointer to the domain for mesh component for the domain associated with the computational node.

Todo

Change this to allocatable???

Definition at line 718 of file types.f90.

7.166.2.5 INTEGER(INTG),allocatable TYPES::DECOMPOSITION_TYPE::ELEMENT_- DOMAIN

ELEMENT_DOMAIN(ne). The domain number that the ne'th global element is in for the decomposition.
Note: the domain numbers start at 0 and go up to the NUMBER_OF_DOMAINS-1.

Definition at line 716 of file types.f90.

7.166.2.6 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::GLOBAL_NUMBER

The global number of the domain decomposition in the list of domain decompositions for a particular mesh.
Definition at line 708 of file types.f90.

7.166.2.7 TYPE(MESH_TYPE),pointer TYPES::DECOMPOSITION_TYPE::MESH

A pointer to the mesh for this decomposition.
Definition at line 711 of file types.f90.

**7.166.2.8 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::MESH_COMPONENT_-
NUMBER**

The component number (index) of the mesh component that this decomposition belongs to (i.e., was generated from).
Definition at line 712 of file types.f90.

7.166.2.9 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::NUMBER_OF_DOMAINS

The number of domains that this decomposition contains.
Definition at line 714 of file types.f90.

**7.166.2.10 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::NUMBER_OF_EDGES_-
CUT**

For automatically calcualted decompositions, the number of edges of the mesh dual graph that were cut for the composition. It provides an indication of the optimally of the automatic decomposition.
Definition at line 715 of file types.f90.

**7.166.2.11 TYPE(DECOMPOSITION_TOPOLOGY_TYPE),pointer
TYPES::DECOMPOSITION_TYPE::TOPOLOGY**

A pointer to the topology for this decomposition.
Definition at line 717 of file types.f90.

7.166.2.12 INTEGER(INTG) TYPES::DECOMPOSITION_TYPE::USER_NUMBER

The user defined identifier for the domain decomposition. The user number must be unique.
Definition at line 707 of file types.f90.

7.167 TYPES::DECOMPOSITIONS_TYPE Struct Reference

Contains information on the domain decompositions defined on a mesh.

Collaboration diagram for TYPES::DECOMPOSITIONS_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer MESH
A pointer to the mesh.
- INTEGER(INTG) NUMBER_OF_DECOMPOSITIONS
The number of decompositions defined on the mesh.
- TYPE(DECOMPOSITION_PTR_TYPE), pointer DECOMPOSITIONS
DECOMPOSITIONS(decomposition_idx). The array of pointers to the domain decompositions.

7.167.1 Detailed Description

Contains information on the domain decompositions defined on a mesh.

Definition at line 727 of file types.f90.

7.167.2 Member Data Documentation

7.167.2.1 TYPE(DECOMPOSITION_PTR_TYPE),pointer TYPES::DECOMPOSITIONS_- TYPE::DECOMPOSITIONS

DECOMPOSITIONS(decomposition_idx). The array of pointers to the domain decompositions.

Definition at line 730 of file types.f90.

7.167.2.2 TYPE(MESH_TYPE),pointer TYPES::DECOMPOSITIONS_TYPE::MESH

A pointer to the mesh.

Definition at line 728 of file types.f90.

7.167.2.3 INTEGER(INTG) TYPES::DECOMPOSITIONS_TYPE::NUMBER_OF_- DECOMPOSITIONS

The number of decompositions defined on the mesh.

Definition at line 729 of file types.f90.

7.168 TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE Struct Reference

Contains information for a [CMISS](#) distributed matrix.

Collaboration diagram for TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE:

Public Attributes

- TYPE([DISTRIBUTED_MATRIX_TYPE](#)), pointer [DISTRIBUTED_MATRIX](#)
A pointer to the distributed matrix.
- INTEGER(INTG) [BASE_TAG_NUMBER](#)
The base number for the MPI tag numbers that will be used to communicate the distributed matrix data amongst the domains. The base tag number can be thought of as the identification number for the distributed matrix object.
- TYPE([MATRIX_TYPE](#)), pointer [MATRIX](#)
A pointer to the matrix to store the rows corresponding to this domain.

7.168.1 Detailed Description

Contains information for a [CMISS](#) distributed matrix.

Definition at line 508 of file types.f90.

7.168.2 Member Data Documentation

7.168.2.1 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE::BASE_TAG_NUMBER

The base number for the MPI tag numbers that will be used to communicate the distributed matrix data amongst the domains. The base tag number can be thought of as the identification number for the distributed matrix object.

Definition at line 510 of file types.f90.

7.168.2.2 TYPE(DISTRIBUTED_MATRIX_TYPE),pointer TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE::DISTRIBUTED_MATRIX

A pointer to the distributed matrix.

Definition at line 509 of file types.f90.

7.168.2.3 TYPE(MATRIX_TYPE),pointer TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE::MATRIX

A pointer to the matrix to store the rows corresponding to this domain.

Definition at line 511 of file types.f90.

7.169 TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE Struct Reference

Contains information for a PETSc distributed matrix.

Collaboration diagram for TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE:

Public Attributes

- TYPE(DISTRIBUTED_MATRIX_TYPE), pointer DISTRIBUTED_MATRIX
A pointer to the distributed matrix.
- INTEGER(INTG) M
The number of local rows in the PETSc matrix.
- INTEGER(INTG) N
The number of local columns in the PETSc matrix.
- INTEGER(INTG) GLOBAL_M
The number of global rows in the PETSc matrix.
- INTEGER(INTG) GLOBAL_N
The number of global columns in the PETSc matrix.
- INTEGER(INTG) STORAGE_TYPE
The storage type (sparsity) of the PETSc matrix.
- INTEGER(INTG) NUMBER_NON_ZEROS
The number of non-zeros in the PETSc matrix.
- INTEGER(INTG) DATA_SIZE
The size of the allocated data in the PETSc matrix.
- INTEGER(INTG) MAXIMUM_COLUMN_INDICES_PER_ROW
The maximum number of column indices for the rows.
- INTEGER(INTG), allocatable DIAGONAL_NUMBER_NON_ZEROS
DIAGONAL_NUMBER_NON_ZEROS(i). The number of non-zeros in the diagonal part of the the i'th row.
- INTEGER(INTG), allocatable OFFDIAGONAL_NUMBER_NON_ZEROS
OFFDIAGONAL_NUMBER_NON_ZEROS(i). The number of non-zeros in the off diagonal part of the the i'th row.
- INTEGER(INTG), allocatable ROW_INDICES
ROW_INDICES(i). The row indices for the matrix.
- INTEGER(INTG), allocatable COLUMN_INDICES
COLUMN_INDICES(i). The column indices for the matrix.
- INTEGER(INTG), allocatable GLOBAL_ROW_NUMBERS

GLOBAL_ROW_NUMBERS(i). The PETSc global row number corresponding to the i'th local row number.

- REAL(DP), allocatable **DATA_DP**

DATA_DP(i). The real data for the matrix.

- TYPE([PETSC_ISLOCALTOGLOBALMAPPING_TYPE](#)) **ISLTGMAPPING**

The local to global mapping for the vector.

- TYPE([PETSC_MAT_TYPE](#)) **MATRIX**

The PETSc matrix.

7.169.1 Detailed Description

Contains information for a PETSc distributed matrix.

Definition at line 515 of file types.f90.

7.169.2 Member Data Documentation

7.169.2.1 INTEGER(INTG),allocatable **TYPES::DISTRIBUTED_MATRIX_PETSC_-TYPE::COLUMN_INDICES**

COLUMN_INDICES(i). The column indices for the matrix.

Definition at line 528 of file types.f90.

7.169.2.2 REAL(DP),allocatable **TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::DATA_DP**

DATA_DP(i). The real data for the matrix.

Definition at line 530 of file types.f90.

7.169.2.3 INTEGER(INTG) **TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::DATA_SIZE**

The size of the allocated data in the PETSc matrix.

Definition at line 523 of file types.f90.

7.169.2.4 INTEGER(INTG),allocatable **TYPES::DISTRIBUTED_MATRIX_PETSC_-TYPE::DIAGONAL_NUMBER_NON_ZEROS**

DIAGONAL_NUMBER_NON_ZEROS(i). The number of non-zeros in the diagonal part of the the i'th row.

Definition at line 525 of file types.f90.

7.169.2.5 TYPE(**DISTRIBUTED_MATRIX_TYPE**),pointer **TYPES::DISTRIBUTED_-MATRIX_PETSC_TYPE::DISTRIBUTED_MATRIX**

A pointer to the distributed matrix.

Definition at line 516 of file types.f90.

7.169.2.6 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::GLOBAL_M

The number of global rows in the PETSc matrix.

Definition at line 519 of file types.f90.

7.169.2.7 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::GLOBAL_N

The number of global columns in the PETSc matrix.

Definition at line 520 of file types.f90.

7.169.2.8 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_MATRIX_PETSC_- TYPE::GLOBAL_ROW_NUMBERS

GLOBAL_ROW_NUMBERS(i). The PETSc global row number corresponding to the i'th local row number.

Definition at line 529 of file types.f90.

7.169.2.9 TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::ISLTGMAPPING

The local to global mapping for the vector.

Definition at line 531 of file types.f90.

7.169.2.10 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::M

The number of local rows in the PETSc matrix.

Definition at line 517 of file types.f90.

7.169.2.11 TYPE(PETSC_MAT_TYPE) TYPES::DISTRIBUTED_MATRIX_PETSC_- TYPE::MATRIX

The PETSc matrix.

Definition at line 532 of file types.f90.

7.169.2.12 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::MAXIMUM_- COLUMN_INDICES_PER_ROW

The maximum number of column indices for the rows.

Definition at line 524 of file types.f90.

7.169.2.13 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::N

The number of local columns in the PETSc matrix.

Definition at line 518 of file types.f90.

**7.169.2.14 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::NUMBER_-
NON_ZEROS**

The number of non-zeros in the PETSc matrix.

Definition at line 522 of file types.f90.

**7.169.2.15 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_MATRIX_PETSC_-
TYPE::OFFDIAGONAL_NUMBER_NON_ZEROS**

OFFDIAGONAL_NUMBER_NON_ZEROS(i). The number of non-zeros in the off diagonal part of the
the i'th row.

Definition at line 526 of file types.f90.

**7.169.2.16 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_MATRIX_PETSC_-
TYPE::ROW_INDICES**

ROW_INDICES(i). The row indices for the matrix.

Definition at line 527 of file types.f90.

**7.169.2.17 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE::STORAGE_-
TYPE**

The storage type (sparsity) of the PETSc matrix.

Definition at line 521 of file types.f90.

7.170 TYPES::DISTRIBUTED_MATRIX_TYPE Struct Reference

Contains the information for a matrix that is distributed across a number of domains.

Collaboration diagram for TYPES::DISTRIBUTED_MATRIX_TYPE:

Public Attributes

- LOGICAL [MATRIX_FINISHED](#)
Is .TRUE. if the distributed matrix has finished being created, .FALSE. if not.
- INTEGER(INTG) [LIBRARY_TYPE](#)
The library of the distributed matrix.
- INTEGER(INTG) [GHOSTING_TYPE](#)
The ghosting type.
- TYPE([DOMAIN_MAPPING_TYPE](#)), pointer [ROW_DOMAIN_MAPPING](#)
The pointer for the domain mapping that identifies how the matrix rows are distributed amongst the domains.
- TYPE([DOMAIN_MAPPING_TYPE](#)), pointer [COLUMN_DOMAIN_MAPPING](#)
The pointer for the domain mapping that identifies how the matrix columns are distributed amongst the domains.
- INTEGER(INTG) [DATA_TYPE](#)
The type of data for the distributed matrix.
- TYPE([DISTRIBUTED_MATRIX_CMISS_TYPE](#)), pointer [CMISS](#)
A pointer to the [CMISS](#) distributed matrix information.
- TYPE([DISTRIBUTED_MATRIX_PETSC_TYPE](#)), pointer [PETSC](#)
A pointer to the PETSc distributed matrix information.

7.170.1 Detailed Description

Contains the information for a matrix that is distributed across a number of domains.

Definition at line 536 of file types.f90.

7.170.2 Member Data Documentation

7.170.2.1 TYPE([DISTRIBUTED_MATRIX_CMISS_TYPE](#)),pointer [TYPES::DISTRIBUTED_MATRIX_TYPE::CMISS](#)

A pointer to the [CMISS](#) distributed matrix information.

Definition at line 543 of file types.f90.

**7.170.2.2 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DISTRIBUTED_MATRIX_-
TYPE::COLUMN_DOMAIN_MAPPING**

The pointer for the domain mapping that identifies how the matrix columns are distributed amongst the domains.

Definition at line 541 of file types.f90.

7.170.2.3 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_TYPE::DATA_TYPE

The type of data for the distributed matrix.

See also:

DISTRIBUTED_MATRIX_VECTOR_DataTypes

Definition at line 542 of file types.f90.

7.170.2.4 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_TYPE::GHOSTING_TYPE

The ghosting type.

See also:

DISTRIBUTED_MATRIX_VECTOR_GhostingTypes,DISTRIBUTED_MATRIX_VECTOR

Definition at line 539 of file types.f90.

7.170.2.5 INTEGER(INTG) TYPES::DISTRIBUTED_MATRIX_TYPE::LIBRARY_TYPE

The library of the distributed matrix.

See also:

DISTRIBUTED_MATRIX_VECTOR_LibraryTypes,DISTRIBUTED_MATRIX_VECTOR

Definition at line 538 of file types.f90.

7.170.2.6 LOGICAL TYPES::DISTRIBUTED_MATRIX_TYPE::MATRIX_FINISHED

Is .TRUE. if the distributed matrix has finished being created, .FALSE. if not.

Definition at line 537 of file types.f90.

**7.170.2.7 TYPE(DISTRIBUTED_MATRIX_PETSC_TYPE),pointer
TYPES::DISTRIBUTED_MATRIX_TYPE::PETSC**

A pointer to the PETSc distributed matrix information.

Definition at line 544 of file types.f90.

**7.170.2.8 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DISTRIBUTED_MATRIX_-
TYPE::ROW_DOMAIN_MAPPING**

The pointer for the domain mapping that identifies how the matrix rows are distributed amongst the domains.

Definition at line 540 of file types.f90.

7.171 TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE Struct Reference

Contains information for a [CMISS](#) distributed vector.

Collaboration diagram for TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE:

Public Attributes

- TYPE([DISTRIBUTED_VECTOR_TYPE](#)), pointer [DISTRIBUTED_VECTOR](#)
A pointer to the distributed vector.
- INTEGER(INTG) [BASE_TAG_NUMBER](#)
The base number for the MPI tag numbers that will be used to communicate the distributed vector data amongst the domains. The base tag number can be thought of as the identification number for the distributed vector object.
- INTEGER(INTG) [N](#)
The size of the distributed vector.
- INTEGER(INTG) [DATA_SIZE](#)
The size of the distributed vector that is held locally by the domain.
- INTEGER(INTG), allocatable [DATA_INTG](#)
DATA_INTG(i). The integer data for an integer distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.
- REAL(DP), allocatable [DATA_DP](#)
DATA_DP(i). The real data for a double precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.
- REAL(SP), allocatable [DATA_SP](#)
DATA_SP(i). The real data for a single precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.
- LOGICAL, allocatable [DATA_L](#)
DATA_L(i). The logical data for a logical distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.
- TYPE([DISTRIBUTED_VECTOR_TRANSFER_TYPE](#)), allocatable [TRANSFERS](#)
TRANSFERS(adjacent_domain_idx). The transfer information for the adjacent_domain_idx'th adjacent domain to this domain.

7.171.1 Detailed Description

Contains information for a [CMISS](#) distributed vector.

Definition at line 473 of file types.f90.

7.171.2 Member Data Documentation

7.171.2.1 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::BASE_TAG - NUMBER

The base number for the MPI tag numbers that will be used to communicate the distributed vector data amongst the domains. The base tag number can be thought of as the identification number for the distributed vector object.

Definition at line 475 of file types.f90.

7.171.2.2 REAL(DP),allocatable TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DATA_DP

DATA_DP(i). The real data for a double precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 479 of file types.f90.

7.171.2.3 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DATA_INTG

DATA_INTG(i). The integer data for an integer distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 478 of file types.f90.

7.171.2.4 LOGICAL,allocatable TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DATA_L

DATA_L(i). The logical data for a logical distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 481 of file types.f90.

7.171.2.5 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DATA_SIZE

The size of the distributed vector that is held locally by the domain.

Definition at line 477 of file types.f90.

7.171.2.6 REAL(SP),allocatable TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DATA_SP

DATA_SP(i). The real data for a single precision real distributed vector. The i'th component contains the data for the i'th local number of distributed vector data on the domain.

Definition at line 480 of file types.f90.

7.171.2.7 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::DISTRIBUTED_VECTOR

A pointer to the distributed vector.

Definition at line 474 of file types.f90.

7.171.2.8 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::N

The size of the distributed vector.

Definition at line 476 of file types.f90.

**7.171.2.9 TYPE(DISTRIBUTED_VECTOR_TRANSFER_TYPE),allocatable
TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE::TRANSFERS**

TRANSFERS(adjacent_domain_idx). The transfer information for the adjacent_domain_idx'th adjacent domain to this domain.

Definition at line 482 of file types.f90.

7.172 TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE Struct Reference

Contains information for a PETSc distributed vector.

Collaboration diagram for TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE:

Public Attributes

- TYPE(DISTRIBUTED_VECTOR_TYPE), pointer DISTRIBUTED_VECTOR
A pointer to the distributed vector.
- INTEGER(INTG) N
The number of local components in the vector.
- INTEGER(INTG) GLOBAL_N
The number of global components in the vector.
- INTEGER(INTG) DATA_SIZE
The size of the distributed vector that is held locally by the domain.
- INTEGER(INTG), allocatable GLOBAL_NUMBERS
GLOBAL_NUMBERS(i). The PETSc global number corresponding to the i'th local number.
- TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE) ISLTGMAPPING
The local to global mapping for the vector.
- TYPE(PETSC_VEC_TYPE) VECTOR
The PETSc vector.

7.172.1 Detailed Description

Contains information for a PETSc distributed vector.

Definition at line 486 of file types.f90.

7.172.2 Member Data Documentation

7.172.2.1 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE::DATA_SIZE

The size of the distributed vector that is held locally by the domain.

Definition at line 490 of file types.f90.

7.172.2.2 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE::DISTRIBUTED_VECTOR

A pointer to the distributed vector.

Definition at line 487 of file types.f90.

7.172.2.3 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE::GLOBAL_N

The number of global components in the vector.

Definition at line 489 of file types.f90.

**7.172.2.4 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_VECTOR_PETSC_-
TYPE::GLOBAL_NUMBERS**

GLOBAL_NUMBERS(i). The PETSc global number corresponding to the i'th local number.

Definition at line 491 of file types.f90.

**7.172.2.5 TYPE(PETSC_ISLOCALTOGLOBALMAPPING_TYPE)
TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE::ISLTGMAPPING**

The local to global mapping for the vector.

Definition at line 492 of file types.f90.

7.172.2.6 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE::N

The number of local components in the vector.

Definition at line 488 of file types.f90.

**7.172.2.7 TYPE(PETSC_VEC_TYPE) TYPES::DISTRIBUTED_VECTOR_PETSC_-
TYPE::VECTOR**

The PETSc vector.

Definition at line 493 of file types.f90.

7.173 TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE Struct Reference

Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.

Collaboration diagram for TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE:

Public Attributes

- TYPE(DISTRIBUTED_VECTOR_CMISS_TYPE), pointer CMISS_VECTOR
The pointer to the CMISS distributed vector object for this transfer information.
- INTEGER(INTG) DATA_TYPE
The data type of the distributed vector. This is "inherited" from the distributed vector.
- INTEGER(INTG) SEND_BUFFER_SIZE
The size of the buffer to send distributed vector data from the current domain to the adjacent domain.
- INTEGER(INTG) RECEIVE_BUFFER_SIZE
The size of the buffer to receive distributed vector data from the adjacent domain to the current domain.
- INTEGER(INTG) SEND_TAG_NUMBER
The MPI tag number for the data sending from the current domain to the adjacent domain. It is calculated as an offset from the base tag number of the distributed vector.
- INTEGER(INTG) RECEIVE_TAG_NUMBER
The MPI tag number for the data receiving from the adjacent domain to the current domain. It is calculated as an offset from the base tag number of the distributed vector.
- INTEGER(INTG) MPI_SEND_REQUEST
The MPI request pointer for sending data from the current domain to the adjacent domain.
- INTEGER(INTG) MPI_RECEIVE_REQUEST
The MPI request pointer for sending data from the adjacent domain to the current domain.
- INTEGER(INTG), allocatable SEND_BUFFER_INTG
The integer buffer for sending the distributed integer vector data from the current domain to the adjacent domain.
- REAL(DP), allocatable SEND_BUFFER_DP
The double precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.
- REAL(SP), allocatable SEND_BUFFER_SP
The single precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.
- LOGICAL, allocatable SEND_BUFFER_L
The logical buffer for sending the distributed logical vector data from the current domain to the adjacent domain.

- INTEGER(INTG), allocatable **RECEIVE_BUFFER_INTG**

The integer buffer for receiving the distributed integer vector data from the adjacent domain to the current domain.

- REAL(DP), allocatable **RECEIVE_BUFFER_DP**

The double precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

- REAL(SP), allocatable **RECEIVE_BUFFER_SP**

The single precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

- LOGICAL, allocatable **RECEIVE_BUFFER_L**

The logical buffer for receiving the distributed logical vector data from the adjacent domain to the current domain.

7.173.1 Detailed Description

Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.

Definition at line 453 of file types.f90.

7.173.2 Member Data Documentation

7.173.2.1 **TYPE(DISTRIBUTED_VECTOR_CMISS_TYPE),pointer TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::CMISS_VECTOR**

The pointer to the **CMISS** distributed vector object for this transfer information.

Definition at line 454 of file types.f90.

7.173.2.2 **INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::DATA_TYPE**

The data type of the distributed vector. This is "inherited" from the distributed vector.

Definition at line 455 of file types.f90.

7.173.2.3 **INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::MPI_RECEIVE_REQUEST**

The MPI request pointer for sending data from the adjacent domain to the current domain.

Definition at line 461 of file types.f90.

7.173.2.4 **INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::MPI_SEND_REQUEST**

The MPI request pointer for sending data from the current domain to the adjacent domain.

Definition at line 460 of file types.f90.

7.173.2.5 REAL(DP),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_BUFFER_DP

The double precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

Definition at line 467 of file types.f90.

7.173.2.6 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_BUFFER_INTG

The integer buffer for receiving the distributed integer vector data from the adjacent domain to the current domain.

Definition at line 466 of file types.f90.

7.173.2.7 LOGICAL,allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_BUFFER_L

The logical buffer for receiving the distributed logical vector data from the adjacent domain to the current domain.

Definition at line 469 of file types.f90.

7.173.2.8 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_BUFFER_SIZE

The size of the buffer to receive distributed vector data from the adjacent domain to the current domain.

Definition at line 457 of file types.f90.

7.173.2.9 REAL(SP),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_BUFFER_SP

The single precision real buffer for receiving the distributed real vector data from the adjacent domain to the current domain.

Definition at line 468 of file types.f90.

7.173.2.10 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_- TYPE::RECEIVE_TAG_NUMBER

The MPI tag number for the data receiving from the adjacent domain to the current domain. It is calculated as an offset from the base tag number of the distributed vector.

Definition at line 459 of file types.f90.

**7.173.2.11 REAL(DP),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_-
TYPE::SEND_BUFFER_DP**

The double precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.

Definition at line 463 of file types.f90.

**7.173.2.12 INTEGER(INTG),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_-
TYPE::SEND_BUFFER_INTG**

The integer buffer for sending the distributed integer vector data from the current domain to the adjacent domain.

Definition at line 462 of file types.f90.

**7.173.2.13 LOGICAL,allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_-
TYPE::SEND_BUFFER_L**

The logical buffer for sending the distributed logical vector data from the current domain to the adjacent domain.

Definition at line 465 of file types.f90.

**7.173.2.14 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::SEND_-
BUFFER_SIZE**

The size of the buffer to send distributed vector data from the current domain to the adjacent domain.

Definition at line 456 of file types.f90.

**7.173.2.15 REAL(SP),allocatable TYPES::DISTRIBUTED_VECTOR_TRANSFER_-
TYPE::SEND_BUFFER_SP**

The single precision real buffer for sending the distributed real vector data from the current domain to the adjacent domain.

Definition at line 464 of file types.f90.

**7.173.2.16 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE::SEND_-
TAG_NUMBER**

The MPI tag number for the data sending from the current domain to the adjacent domain. It is calculated as an offset from the base tag number of the distributed vector.

Definition at line 458 of file types.f90.

7.174 TYPES::DISTRIBUTED_VECTOR_TYPE Struct Reference

Contains the information for a vector that is distributed across a number of domains.

Collaboration diagram for TYPES::DISTRIBUTED_VECTOR_TYPE:

Public Attributes

- LOGICAL [VECTOR_FINISHED](#)
!<Is .TRUE. if the distributed vector has finished being created, .FALSE. if not.
- INTEGER(INTG) [LIBRARY_TYPE](#)
The format of the distributed vector.
- INTEGER(INTG) [GHOSTING_TYPE](#)
The ghosting type.
- TYPE([DOMAIN_MAPPING_TYPE](#)), pointer [DOMAIN_MAPPING](#)
The pointer for the domain mapping that identifies how the vector is distributed amongst the domains.
- INTEGER(INTG) [DATA_TYPE](#)
The type of data for the distributed vector.
- TYPE([DISTRIBUTED_VECTOR_CMISS_TYPE](#)), pointer [CMISS](#)
A pointer to the [CMISS](#) distributed vector information.
- TYPE([DISTRIBUTED_VECTOR_PETSC_TYPE](#)), pointer [PETSC](#)
A pointer to the PETSc distributed vector information.

7.174.1 Detailed Description

Contains the information for a vector that is distributed across a number of domains.

Definition at line 497 of file types.f90.

7.174.2 Member Data Documentation

7.174.2.1 TYPE([DISTRIBUTED_VECTOR_CMISS_TYPE](#)),pointer TYPES::DISTRIBUTED_VECTOR_TYPE::CMISS

A pointer to the [CMISS](#) distributed vector information.

Definition at line 503 of file types.f90.

7.174.2.2 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TYPE::DATA_TYPE

The type of data for the distributed vector.

See also:

[DISTRIBUTED_MATRIX_VECTOR_DataTypes](#)

Definition at line 502 of file types.f90.

**7.174.2.3 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DISTRIBUTED_VECTOR_-
TYPE::DOMAIN_MAPPING**

The pointer for the domain mapping that identifies how the vector is distributed amongst the domains.

Definition at line 501 of file types.f90.

7.174.2.4 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TYPE::GHOSTING_TYPE

The ghosting type.

See also:

[DISTRIBUTED_MATRIX_VECTOR_GhostingTypes](#),[DISTRIBUTED_MATRIX_VECTOR](#)

Definition at line 500 of file types.f90.

7.174.2.5 INTEGER(INTG) TYPES::DISTRIBUTED_VECTOR_TYPE::LIBRARY_TYPE

The format of the distributed vector.

See also:

[DISTRIBUTED_MATRIX_VECTOR_LibraryTypes](#),[DISTRIBUTED_MATRIX_VECTOR](#)

Definition at line 499 of file types.f90.

**7.174.2.6 TYPE(DISTRIBUTED_VECTOR_PETSC_TYPE),pointer
TYPES::DISTRIBUTED_VECTOR_TYPE::PETSC**

A pointer to the PETSc distributed vector information.

Definition at line 504 of file types.f90.

7.174.2.7 LOGICAL TYPES::DISTRIBUTED_VECTOR_TYPE::VECTOR_FINISHED

!<Is .TRUE. if the distributed vector has finished being created, .FALSE. if not.

Definition at line 498 of file types.f90.

7.175 TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE Struct Reference

Contains the information on an adjacent domain to a domain in a domain mapping.

Public Attributes

- INTEGER(INTG) [DOMAIN_NUMBER](#)

The number of the domain that is adjacent to a domain in a mapping.

- INTEGER(INTG) [NUMBER_OF_SEND_GHOSTS](#)

The number of ghost locals in the current domain that are to be sent to this domain.

- INTEGER(INTG) [NUMBER_OF_RECEIVE_GHOSTS](#)

The number of ghost locals in the current domain that are to be received from this domain.

- INTEGER(INTG), allocatable [LOCAL_GHOST_SEND_INDICES](#)

LOCAL_GHOST_SEND_INDICES(i). The local numbers of the ghosts in the current domain that are to be sent to this domain.

- INTEGER(INTG), allocatable [LOCAL_GHOST_RECEIVE_INDICES](#)

LOCAL_GHOST_RECEIVE_INDICES(i). The local numbers of the ghosts in the current domain that are to be received from this domain.

7.175.1 Detailed Description

Contains the information on an adjacent domain to a domain in a domain mapping.

Definition at line 594 of file types.f90.

7.175.2 Member Data Documentation

7.175.2.1 INTEGER(INTG) TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE::DOMAIN_NUMBER

The number of the domain that is adjacent to a domain in a mapping.

Definition at line 595 of file types.f90.

7.175.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE::LOCAL_GHOST_RECEIVE_INDICES

LOCAL_GHOST_RECEIVE_INDICES(i). The local numbers of the ghosts in the current domain that are to be received from this domain.

Definition at line 599 of file types.f90.

**7.175.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN_ADJACENT_DOMAIN_-
TYPE::LOCAL_GHOST_SEND_INDICES**

LOCAL_GHOST_SEND_INDICES(i). The local numbers of the ghosts in the current domain that are to be sent to this domain.

Definition at line 598 of file types.f90.

**7.175.2.4 INTEGER(INTG) TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE::NUMBER_-
OF_RECEIVE_GHOSTS**

The number of ghost locals in the current domain that are to be received from this domain.

Definition at line 597 of file types.f90.

**7.175.2.5 INTEGER(INTG) TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE::NUMBER_-
OF_SEND_GHOSTS**

The number of ghost locals in the current domain that are to be sent to this domain.

Definition at line 596 of file types.f90.

7.176 TYPES::DOMAIN_DOFS_TYPE Struct Reference

Contains information on the degrees-of-freedom (dofs) for a domain.

Collaboration diagram for TYPES::DOMAIN_DOFS_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN

A pointer to the domain.

- INTEGER(INTG) NUMBER_OF_DOFS

The number of degrees-of-freedom (excluding ghost dofs) in the domain.

- INTEGER(INTG) TOTAL_NUMBER_OF_DOFS

The total number of degrees-of-freedom (including ghost dofs) in the domain.

- INTEGER(INTG), allocatable DOF_INDEX

DOF_INDEX(i,ny). The index for the ny'th degree-of-freedom. When i=1 DOF_INDEX will give the global derivative number (nk) associated with the dof. When i=2 DOF_INDEX will give the local node number (np) associated with the dof.

7.176.1 Detailed Description

Contains information on the degrees-of-freedom (dofs) for a domain.

Definition at line 346 of file types.f90.

7.176.2 Member Data Documentation

7.176.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN_DOFS_TYPE::DOF_INDEX

DOF_INDEX(i,ny). The index for the ny'th degree-of-freedom. When i=1 DOF_INDEX will give the global derivative number (nk) associated with the dof. When i=2 DOF_INDEX will give the local node number (np) associated with the dof.

Definition at line 350 of file types.f90.

7.176.2.2 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_DOFS_TYPE::DOMAIN

A pointer to the domain.

Definition at line 347 of file types.f90.

7.176.2.3 INTEGER(INTG) TYPES::DOMAIN_DOFS_TYPE::NUMBER_OF_DOFS

The number of degrees-of-freedom (excluding ghost dofs) in the domain.

Definition at line 348 of file types.f90.

7.176.2.4 INTEGER(INTG) TYPES::DOMAIN_DOFS_TYPE::TOTAL_NUMBER_OF_DOFS

The total number of degrees-of-freedom (including ghost dofs) in the domain.

Definition at line 349 of file types.f90.

7.177 TYPES::DOMAIN_ELEMENT_TYPE Struct Reference

Contains the information for an element in a domain.

Collaboration diagram for TYPES::DOMAIN_ELEMENT_TYPE:

Public Attributes

- INTEGER(INTG) **NUMBER**
The local element number in the domain.
- TYPE(BASIS_TYPE), pointer **BASIS**
A pointer to the basis function for the element.
- INTEGER(INTG), allocatable **ELEMENT_NODES**
ELEMENT_NODES(nn). The local node number in the domain of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).
- INTEGER(INTG), allocatable **ELEMENT_DERIVATIVES**
ELEMENT_DERIVATIVES(nk,nn). The global derivative number of the local derivative nk for the local node nn in the element. Old CMISS name NKJE(nk,nn,nj,ne).

7.177.1 Detailed Description

Contains the information for an element in a domain.

Definition at line 396 of file types.f90.

7.177.2 Member Data Documentation

7.177.2.1 TYPE(BASIS_TYPE),pointer TYPES::DOMAIN_ELEMENT_TYPE::BASIS

A pointer to the basis function for the element.

Definition at line 398 of file types.f90.

7.177.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN_ELEMENT_TYPE::ELEMENT_DERIVATIVES

ELEMENT_DERIVATIVES(nk,nn). The global derivative number of the local derivative nk for the local node nn in the element. Old CMISS name NKJE(nk,nn,nj,ne).

Definition at line 400 of file types.f90.

7.177.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN_ELEMENT_TYPE::ELEMENT_NODES

ELEMENT_NODES(nn). The local node number in the domain of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).

Definition at line 399 of file types.f90.

7.177.2.4 INTEGER(INTG) TYPES::DOMAIN_ELEMENT_TYPE::NUMBER

The local element number in the domain.

Definition at line 397 of file types.f90.

7.178 TYPES::DOMAIN_ELEMENTS_TYPE Struct Reference

Contains the topology information for the elements of a domain.

Collaboration diagram for TYPES::DOMAIN_ELEMENTS_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
The pointer to the domain for this elements topology information.
- INTEGER(INTG) NUMBER_OF_ELEMENTS
The number of elements (excluding ghost elements) in this domain topology.
- INTEGER(INTG) TOTAL_NUMBER_OF_ELEMENTS
The total number of elements (including ghost elements) in this domain topology.
- TYPE(DOMAIN_ELEMENT_TYPE), pointer ELEMENTS
ELEMENTS(ne). The pointer to the array of topology information for the elements of this domain. ELEMENTS(ne) contains the topological information for the ne'th local elements of the domain.
- INTEGER(INTG) MAXIMUM_NUMBER_OF_ELEMENT_PARAMETERS
The maximum number of element parameters (ns) for all the elements in the domain.

7.178.1 Detailed Description

Contains the topology information for the elements of a domain.

Definition at line 404 of file types.f90.

7.178.2 Member Data Documentation

7.178.2.1 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_ELEMENTS_TYPE::DOMAIN

The pointer to the domain for this elements topology information.

Definition at line 405 of file types.f90.

7.178.2.2 TYPE(DOMAIN_ELEMENT_TYPE),pointer TYPES::DOMAIN_ELEMENTS_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of topology information for the elements of this domain. ELEMENTS(ne) contains the topological information for the ne'th local elements of the domain.

Todo

Change this to allocatable???

Definition at line 408 of file types.f90.

7.178.2.3 INTEGER(INTG) TYPES::DOMAIN_ELEMENTS_TYPE::MAXIMUM_NUMBER_OF_ELEMENT_PARAMETERS

The maximum number of element parameters (ns) for all the elements in the domain.

Definition at line 409 of file types.f90.

7.178.2.4 INTEGER(INTG) TYPES::DOMAIN_ELEMENTS_TYPE::NUMBER_OF_ELEMENTS

The number of elements (excluding ghost elements) in this domain topology.

Definition at line 406 of file types.f90.

7.178.2.5 INTEGER(INTG) TYPES::DOMAIN_ELEMENTS_TYPE::TOTAL_NUMBER_OF_ELEMENTS

The total number of elements (including ghost elements) in this domain topology.

Definition at line 407 of file types.f90.

7.179 TYPES::DOMAIN_FACE_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).

Collaboration diagram for TYPES::DOMAIN_FACE_PTR_TYPE:

Public Attributes

- TYPE([DOMAIN_FACE_TYPE](#)), pointer PTR

The pointer to the domain face.

7.179.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).

Definition at line 384 of file types.f90.

7.179.2 Member Data Documentation

7.179.2.1 TYPE(DOMAIN_FACE_TYPE),pointer TYPES::DOMAIN_FACE_PTR_TYPE::PTR

The pointer to the domain face.

Definition at line 385 of file types.f90.

7.180 TYPES::DOMAIN_FACE_TYPE Struct Reference

Contains the information for a face in a domain.

Collaboration diagram for TYPES::DOMAIN_FACE_TYPE:

Public Attributes

- INTEGER(INTG) **NUMBER**
The face number in the domain.
- INTEGER(INTG) **XI_DIRECTION1**
The first xi direction of the face.
- INTEGER(INTG) **XI_DIRECTION2**
The second xi direction of the face.
- TYPE(**BASIS_TYPE**), pointer **BASIS**
A pointer to the basis function for the face.
- INTEGER(INTG), allocatable **NODES_IN_FACE**
NODES_IN_FACE(nn). The local node number in the domain of the nn'th local node in the face. Old CMISS name NPNF(nn,nbf).
- INTEGER(INTG), allocatable **DERIVATIVES_IN_FACE**
DERIVATIVES_IN_FACE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the face.

7.180.1 Detailed Description

Contains the information for a face in a domain.

Definition at line 374 of file types.f90.

7.180.2 Member Data Documentation

7.180.2.1 TYPE(**BASIS_TYPE**),pointer **TYPES::DOMAIN_FACE_TYPE::BASIS**

A pointer to the basis function for the face.

Definition at line 378 of file types.f90.

7.180.2.2 INTEGER(INTG),allocatable **TYPES::DOMAIN_FACE_TYPE::DERIVATIVES_IN_FACE**

DERIVATIVES_IN_FACE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the face.

Definition at line 380 of file types.f90.

7.180.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN_FACE_TYPE::NODES_IN_FACE

NODES_IN_FACE(nn). The local node number in the domain of the nn'th local node in the face. Old CMISS name NPNF(nn,nbf).

Definition at line 379 of file types.f90.

7.180.2.4 INTEGER(INTG) TYPES::DOMAIN_FACE_TYPE::NUMBER

The face number in the domain.

Definition at line 375 of file types.f90.

7.180.2.5 INTEGER(INTG) TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION1

The first xi direction of the face.

Todo

move this to the decomposition face type

Definition at line 376 of file types.f90.

7.180.2.6 INTEGER(INTG) TYPES::DOMAIN_FACE_TYPE::XI_DIRECTION2

The second xi direction of the face.

Todo

move this to the decomposition face type

Definition at line 377 of file types.f90.

7.181 TYPES::DOMAIN_FACES_TYPE Struct Reference

Contains the topology information for the faces of a domain.

Collaboration diagram for TYPES::DOMAIN_FACES_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
The pointer to the domain for this faces topology information.
- INTEGER(INTG) NUMBER_OF_FACES
The number of faces in this domain topology.
- TYPE(DOMAIN_FACE_TYPE), allocatable FACES
FACES(nf). The pointer to the array of topology information for the faces of this domain. FACES(nf) contains the topological information for the nf'th local face of the domain.

7.181.1 Detailed Description

Contains the topology information for the faces of a domain.

Definition at line 389 of file types.f90.

7.181.2 Member Data Documentation

7.181.2.1 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_FACES_TYPE::DOMAIN

The pointer to the domain for this faces topology information.

Definition at line 390 of file types.f90.

7.181.2.2 TYPE(DOMAIN_FACE_TYPE),allocatable TYPES::DOMAIN_FACES_TYPE::FACES

FACES(nf). The pointer to the array of topology information for the faces of this domain. FACES(nf) contains the topological information for the nf'th local face of the domain.

Definition at line 392 of file types.f90.

7.181.2.3 INTEGER(INTG) TYPES::DOMAIN_FACES_TYPE::NUMBER_OF_FACES

The number of faces in this domain topology.

Definition at line 391 of file types.f90.

7.182 TYPES::DOMAIN_GLOBAL_MAPPING_TYPE Struct Reference

Contains the local information for a global mapping number for a domain mapping.

Public Attributes

- INTEGER(INTG) [NUMBER_OF_DOMAINS](#)

The number of domains that the global number is mapped to a local number in.

- INTEGER(INTG), allocatable [LOCAL_NUMBER](#)

LOCAL_NUMBER(domain_idx). The mapped local number for the domain_idx'th domain for the global number.

- INTEGER(INTG), allocatable [DOMAIN_NUMBER](#)

DOMAIN_NUMBER(domain_idx). The domain number for the domain_idx'th domain for which the global number is mapped to a local number.

- INTEGER(INTG), allocatable [LOCAL_TYPE](#)

LOCAL_TYPE(domain_idx). The type of local for the domain_idx'th domain for which the global number is mapped to a local number. The types depend on whether the mapped local number in the domain_idx'th domain is an internal, boundary or ghost local number.

7.182.1 Detailed Description

Contains the local information for a global mapping number for a domain mapping.

Definition at line 603 of file types.f90.

7.182.2 Member Data Documentation

7.182.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN_GLOBAL_MAPPING_- TYPE::DOMAIN_NUMBER

DOMAIN_NUMBER(domain_idx). The domain number for the domain_idx'th domain for which the global number is mapped to a local number.

Definition at line 606 of file types.f90.

7.182.2.2 INTEGER(INTG),allocatable TYPES::DOMAIN_GLOBAL_MAPPING_- TYPE::LOCAL_NUMBER

LOCAL_NUMBER(domain_idx). The mapped local number for the domain_idx'th domain for the global number.

Definition at line 605 of file types.f90.

**7.182.2.3 INTEGER(INTG),allocatable TYPES::DOMAIN_GLOBAL_MAPPING_-
TYPE::LOCAL_TYPE**

LOCAL_TYPE(domain_idx). The type of local for the domain_idx'th domain for which the global number is mapped to a local number. The types depend on whether the mapped local number in the domain_idx'th domain is an internal, boundary or ghost local number.

See also:

[DOMAIN_MAPPINGS::DomainType](#)

Definition at line 607 of file types.f90.

**7.182.2.4 INTEGER(INTG) TYPES::DOMAIN_GLOBAL_MAPPING_TYPE::NUMBER_OF_-
DOMAINS**

The number of domains that the global number is mapped to a local number in.

Definition at line 604 of file types.f90.

7.183 TYPES::DOMAIN_LINE_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DOMAIN_LINE_TYPE](#).

Collaboration diagram for TYPES::DOMAIN_LINE_PTR_TYPE:

Public Attributes

- TYPE([DOMAIN_LINE_TYPE](#)), pointer PTR
A pointer to the domain line.

7.183.1 Detailed Description

A buffer type to allow for an array of pointers to a [DOMAIN_LINE_TYPE](#).

Definition at line 362 of file types.f90.

7.183.2 Member Data Documentation

7.183.2.1 TYPE(DOMAIN_LINE_TYPE),pointer TYPES::DOMAIN_LINE_PTR_TYPE::PTR

A pointer to the domain line.

Definition at line 363 of file types.f90.

7.184 TYPES::DOMAIN_LINE_TYPE Struct Reference

Contains the information for a line in a domain.

Collaboration diagram for TYPES::DOMAIN_LINE_TYPE:

Public Attributes

- INTEGER(INTG) **NUMBER**

The line number in the domain.

- TYPE(**BASIS_TYPE**), pointer **BASIS**

A pointer to the basis function for the line.

- INTEGER(INTG), allocatable **NODES_IN_LINE**

NODES_IN_LINE(nn). The local node number in the domain of the nn'th local node in the line. Old CMISS name NPL(2..5,nj,nl).

- INTEGER(INTG), allocatable **DERIVATIVES_IN_LINE**

DERIVATIVES_IN_LINE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the line. Old CMISS name NPL(4..5,nj,nl).

7.184.1 Detailed Description

Contains the information for a line in a domain.

Definition at line 354 of file types.f90.

7.184.2 Member Data Documentation

7.184.2.1 TYPE(**BASIS_TYPE**),pointer **TYPES::DOMAIN_LINE_TYPE::BASIS**

A pointer to the basis function for the line.

Definition at line 356 of file types.f90.

7.184.2.2 INTEGER(INTG),allocatable **TYPES::DOMAIN_LINE_TYPE::DERIVATIVES_IN_LINE**

DERIVATIVES_IN_LINE(nk,nn). The global derivative number of the local derivative nk for the local node nn in the line. Old CMISS name NPL(4..5,nj,nl).

Definition at line 358 of file types.f90.

7.184.2.3 INTEGER(INTG),allocatable **TYPES::DOMAIN_LINE_TYPE::NODES_IN_LINE**

NODES_IN_LINE(nn). The local node number in the domain of the nn'th local node in the line. Old CMISS name NPL(2..5,nj,nl).

Definition at line 357 of file types.f90.

7.184.2.4 INTEGER(INTG) TYPES::DOMAIN_LINE_TYPE::NUMBER

The line number in the domain.

Definition at line 355 of file types.f90.

7.185 TYPES::DOMAIN_LINES_TYPE Struct Reference

Contains the topology information for the lines of a domain.

Collaboration diagram for TYPES::DOMAIN_LINES_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
The pointer to the domain for this lines topology information.
- INTEGER(INTG) NUMBER_OF_LINES
The number of lines in this domain topology.
- TYPE(DOMAIN_LINE_TYPE), allocatable LINES
LINES(nl). The pointer to the array of topology information for the lines of this domain. LINES(nl) contains the topological information for the nl'th local line of the domain.

7.185.1 Detailed Description

Contains the topology information for the lines of a domain.

Definition at line 367 of file types.f90.

7.185.2 Member Data Documentation

7.185.2.1 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_LINES_TYPE::DOMAIN

The pointer to the domain for this lines topology information.

Definition at line 368 of file types.f90.

7.185.2.2 TYPE(DOMAIN_LINE_TYPE),allocatable TYPES::DOMAIN_LINES_TYPE::LINES

LINES(nl). The pointer to the array of topology information for the lines of this domain. LINES(nl) contains the topological information for the nl'th local line of the domain.

Definition at line 370 of file types.f90.

7.185.2.3 INTEGER(INTG) TYPES::DOMAIN_LINES_TYPE::NUMBER_OF_LINES

The number of lines in this domain topology.

Definition at line 369 of file types.f90.

7.186 TYPES::DOMAIN_MAPPING_TYPE Struct Reference

Contains information on the domain mappings (i.e., local and global numberings).

Collaboration diagram for TYPES::DOMAIN_MAPPING_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_OF_LOCAL](#)
The number of local numbers in the domain excluding ghost numbers.
- INTEGER(INTG) [TOTAL_NUMBER_OF_LOCAL](#)
The total number of local numbers in the domain including ghost numbers.
- INTEGER(INTG), allocatable [NUMBER_OF_DOMAIN_LOCAL](#)
*NUMBER_OF_DOMAIN_LOCAL(domain_no). The total number of locals for domain_no'th domain.
 NOTE: the domain_no goes from 0 to the number of domains-1.*
- INTEGER(INTG) [NUMBER_OF_GLOBAL](#)
The number of global numbers for this mapping.
- INTEGER(INTG) [NUMBER_OF_DOMAINS](#)
The number of domains in this mapping.
- INTEGER(INTG) [NUMBER_OF_INTERNAL](#)
The number of internal numbers in this mapping.
- INTEGER(INTG), allocatable [INTERNAL_LIST](#)
INTERNAL_LIST(i). The list of internal numbers for the mapping. The i'th position gives the i'th local internal number.
- INTEGER(INTG) [NUMBER_OF_BOUNDARY](#)
The number of boundary numbers in this mapping.
- INTEGER(INTG), allocatable [BOUNDARY_LIST](#)
BOUNDARY_LIST(i). The list of boundary numbers for the mapping. The i'th position gives the i'th local boundary number.
- INTEGER(INTG) [NUMBER_OF_GHOST](#)
The number of ghost numbers in this mapping.
- INTEGER(INTG), allocatable [GHOST_LIST](#)
GHOST_LIST(i). The list of ghost numbers for the mapping. The i'th position gives the i'th local ghost number.
- INTEGER(INTG), allocatable [LOCAL_TO_GLOBAL_MAP](#)
LOCAL_TO_GLOBAL_MAP(i). The global number for the i'th local number for the mapping.
- TYPE([DOMAIN_GLOBAL_MAPPING_TYPE](#)), allocatable [GLOBAL_TO_LOCAL_MAP](#)
GLOBAL_TO_LOCAL_MAP(i). The local information for the i'th global number for the mapping.

- INTEGER(INTG) [NUMBER_OF_ADJACENT_DOMAINS](#)
The number of domains that are adjacent to this domain in the mapping.
- INTEGER(INTG), allocatable [ADJACENT_DOMAINS_PTR](#)
ADJACENT_DOMAINS_PTR(domain_no). The pointer to the list of adjacent domains for domain_no. ADJACENT_DOMAINS_PTR(domain_no) gives the starting position in ADJACENT_DOMAINS_LIST for the first adjacent domain number for domain number domain_no. ADJACENT_DOMAINS_PTR(domain_no+1) gives the last+1 position in ADJACENT_DOMAINS_LIST for the last adjacent domain number for domain number domain_no. NOTE: the index for ADJACENT_DOMAINS_PTR varies from 0 to the number of domains+1.
- INTEGER(INTG), allocatable [ADJACENT_DOMAINS_LIST](#)
ADJACENT_DOMAINS_LIST(i). The list of adjacent domains for each domain. The start and end positions for the list for domain number domain_no are given by ADJACENT_DOMAIN_PTR(domain_no) and ADJACENT_DOMAIN_PTR(domain_no+1)-1 respectively.
- TYPE([DOMAIN_ADJACENT_DOMAIN_TYPE](#)), allocatable [ADJACENT_DOMAINS](#)
ADJACENT_DOMAINS(adjacent_domain_idx). The adjacent domain information for the adjacent_domain_idx'th adjacent domain to this domain.

7.186.1 Detailed Description

Contains information on the domain mappings (i.e., local and global numberings).

Definition at line 611 of file types.f90.

7.186.2 Member Data Documentation

7.186.2.1 TYPE([DOMAIN_ADJACENT_DOMAIN_TYPE](#)),allocatable [TYPES::DOMAIN_MAPPING_TYPE::ADJACENT_DOMAINS](#)

ADJACENT_DOMAINS(adjacent_domain_idx). The adjacent domain information for the adjacent_domain_idx'th adjacent domain to this domain.

Definition at line 630 of file types.f90.

7.186.2.2 INTEGER(INTG),allocatable [TYPES::DOMAIN_MAPPING_TYPE::ADJACENT_DOMAINS_LIST](#)

ADJACENT_DOMAINS_LIST(i). The list of adjacent domains for each domain. The start and end positions for the list for domain number domain_no are given by ADJACENT_DOMAIN_PTR(domain_no) and ADJACENT_DOMAIN_PTR(domain_no+1)-1 respectively.

Definition at line 629 of file types.f90.

7.186.2.3 INTEGER(INTG),allocatable [TYPES::DOMAIN_MAPPING_TYPE::ADJACENT_DOMAINS_PTR](#)

ADJACENT_DOMAINS_PTR(domain_no). The pointer to the list of adjacent domains for domain_no. ADJACENT_DOMAINS_PTR(domain_no) gives the starting position in ADJACENT_DOMAINS_-

LIST for the first adjacent domain number for domain number domain_no. ADJACENT_DOMAINS_PTR(domain_no+1) gives the last+1 position in ADJACENT_DOMAINS_LIST for the last adjacent domain number for domain number domain_no. NOTE: the index for ADJACENT_DOMAINS_PTR varies from 0 to the number of domains+1.

Definition at line 628 of file types.f90.

7.186.2.4 INTEGER(INTG),allocatable TYPES::DOMAIN_MAPPING_TYPE::BOUNDARY_LIST

BOUNDARY_LIST(i). The list of boundary numbers for the mapping. The i'th position gives the i'th local boundary number.

Definition at line 620 of file types.f90.

7.186.2.5 INTEGER(INTG),allocatable TYPES::DOMAIN_MAPPING_TYPE::GHOST_LIST

GHOST_LIST(i). The list of ghost numbers for the mapping. The i'th position gives the i'th local ghost number.

Definition at line 622 of file types.f90.

7.186.2.6 TYPE(DOMAIN_GLOBAL_MAPPING_TYPE),allocatable TYPES::DOMAIN_MAPPING_TYPE::GLOBAL_TO_LOCAL_MAP

GLOBAL_TO_LOCAL_MAP(i). The local information for the i'th global number for the mapping.

Definition at line 626 of file types.f90.

7.186.2.7 INTEGER(INTG),allocatable TYPES::DOMAIN_MAPPING_TYPE::INTERNAL_LIST

INTERNAL_LIST(i). The list of internal numbers for the mapping. The i'th position gives the i'th local internal number.

Definition at line 618 of file types.f90.

7.186.2.8 INTEGER(INTG),allocatable TYPES::DOMAIN_MAPPING_TYPE::LOCAL_TO_GLOBAL_MAP

LOCAL_TO_GLOBAL_MAP(i). The global number for the i'th local number for the mapping.

Definition at line 625 of file types.f90.

7.186.2.9 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_ADJACENT_DOMAINS

The number of domains that are adjacent to this domain in the mapping.

Definition at line 627 of file types.f90.

7.186.2.10 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_BOUNDARY

The number of boundary numbers in this mapping.

Definition at line 619 of file types.f90.

7.186.2.11 INTEGER(INTG),allocatable TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_DOMAIN_LOCAL

NUMBER_OF_DOMAIN_LOCAL(domain_no). The total number of locals for domain_no'th domain.
NOTE: the domain_no goes from 0 to the number of domains-1.

Definition at line 614 of file types.f90.

7.186.2.12 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_DOMAINS

The number of domains in this mapping.

Definition at line 616 of file types.f90.

7.186.2.13 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_GHOST

The number of ghost numbers in this mapping.

Definition at line 621 of file types.f90.

7.186.2.14 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_GLOBAL

The number of global numbers for this mapping.

Definition at line 615 of file types.f90.

7.186.2.15 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_INTERNAL

The number of internal numbers in this mapping.

Definition at line 617 of file types.f90.

7.186.2.16 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::NUMBER_OF_LOCAL

The number of local numbers in the domain excluding ghost numbers.

Definition at line 612 of file types.f90.

7.186.2.17 INTEGER(INTG) TYPES::DOMAIN_MAPPING_TYPE::TOTAL_NUMBER_OF_LOCAL

The total number of local numbers in the domain including ghost numbers.

Definition at line 613 of file types.f90.

7.187 TYPES::DOMAIN_MAPPINGS_TYPE Struct Reference

Contains information on the domain decomposition mappings.

Collaboration diagram for TYPES::DOMAIN_MAPPINGS_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
A pointer to the domain decomposition.
- TYPE(DOMAIN_MAPPING_TYPE), pointer ELEMENTS
Pointer to the element mappings for the domain decomposition.
- TYPE(DOMAIN_MAPPING_TYPE), pointer NODES
Pointer to the node mappings for the domain decomposition.
- TYPE(DOMAIN_MAPPING_TYPE), pointer DOFS
Pointer to the dof mappings for the domain decomposition.

7.187.1 Detailed Description

Contains information on the domain decomposition mappings.

Definition at line 634 of file types.f90.

7.187.2 Member Data Documentation

7.187.2.1 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DOMAIN_MAPPINGS_- TYPE::DOFS

Pointer to the dof mappings for the domain decomposition.

Definition at line 638 of file types.f90.

7.187.2.2 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_MAPPINGS_TYPE::DOMAIN

A pointer to the domain decomposition.

Definition at line 635 of file types.f90.

7.187.2.3 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DOMAIN_MAPPINGS_- TYPE::ELEMENTS

Pointer to the element mappings for the domain decomposition.

Definition at line 636 of file types.f90.

**7.187.2.4 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::DOMAIN_MAPPINGS_-
TYPE::NODES**

Pointer to the node mappings for the domain decomposition.

Definition at line 637 of file types.f90.

7.188 TYPES::DOMAIN_NODE_TYPE Struct Reference

Contains the topology information for a local node of a domain.

Public Attributes

- INTEGER(INTG) [LOCAL_NUMBER](#)

The local node number in the domain.

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The corresponding global node number in the mesh of the local node number in the domain.

- INTEGER(INTG) [USER_NUMBER](#)

The corresponding user number for the node.

- INTEGER(INTG) [NUMBER_OF_DERIVATIVES](#)

The number of global derivatives at the node for the domain. Old [CMISS](#) name NKT(nj,np).

- INTEGER(INTG), allocatable [PARTIAL_DERIVATIVE_INDEX](#)

PARTIAL_DERIVATIVE_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).

- INTEGER(INTG), allocatable [DOF_INDEX](#)

DOF_INDEX(nk). The local dof derivative index (ny) in the domain of the nk'th global derivative for the node.

- INTEGER(INTG) [NUMBER_OF_SURROUNDING_ELEMENTS](#)

The number of elements surrounding the node in the domain. Old [CMISS](#) name NENP(np,0,0:nr).

- INTEGER(INTG), pointer [SURROUNDING_ELEMENTS](#)

SURROUNDING_ELEMENTS(nep). The local element number of the nep'th element that is surrounding the node. Old [CMISS](#) name NENP(np,nep,0:nr).

- INTEGER(INTG) [NUMBER_OF_NODE_LINES](#)

The number of lines surrounding the node in the domain.

- INTEGER(INTG), allocatable [NODE_LINES](#)

NODE_LINES(nlp). The local line number of the nlp'th line that is surrounding the node.

7.188.1 Detailed Description

Contains the topology information for a local node of a domain.

Definition at line 413 of file types.f90.

7.188.2 Member Data Documentation

7.188.2.1 INTEGER(INTG),allocatable TYPES::DOMAIN_NODE_TYPE::DOF_INDEX

DOF_INDEX(nk). The local dof derivative index (ny) in the domain of the nk'th global derivative for the node.

Definition at line 419 of file types.f90.

7.188.2.2 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::GLOBAL_NUMBER

The corresponding global node number in the mesh of the local node number in the domain.

Definition at line 415 of file types.f90.

7.188.2.3 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::LOCAL_NUMBER

The local node number in the domain.

Definition at line 414 of file types.f90.

7.188.2.4 INTEGER(INTG),allocatable TYPES::DOMAIN_NODE_TYPE::NODE_LINES

NODE_LINES(nlp). The local line number of the nlp'th line that is surrounding the node.

Definition at line 423 of file types.f90.

7.188.2.5 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::NUMBER_OF_DERIVATIVES

The number of global derivatives at the node for the domain. Old [CMISS](#) name NKT(nj,np).

Definition at line 417 of file types.f90.

7.188.2.6 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::NUMBER_OF_NODE_LINES

The number of lines surrounding the node in the domain.

Definition at line 422 of file types.f90.

7.188.2.7 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::NUMBER_OF_- SURROUNDING_ELEMENTS

The number of elements surrounding the node in the domain. Old [CMISS](#) name NENP(np,0,0:nr).

Definition at line 420 of file types.f90.

7.188.2.8 INTEGER(INTG),allocatable TYPES::DOMAIN_NODE_TYPE::PARTIAL_- DERIVATIVE_INDEX

PARTIAL_DERIVATIVE_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).

Definition at line 418 of file types.f90.

7.188.2.9 INTEGER(INTG),pointer TYPES::DOMAIN_NODE_TYPE::SURROUNDING_ELEMENTS

SURROUNDING_ELEMENTS(nep). The local element number of the nep'th element that is surrounding the node. Old CMISS name NENP(np,nep,0:nr).

Todo

Change this to allocatable.

Definition at line 421 of file types.f90.

7.188.2.10 INTEGER(INTG) TYPES::DOMAIN_NODE_TYPE::USER_NUMBER

The corresponding user number for the node.

Definition at line 416 of file types.f90.

7.189 TYPES::DOMAIN_NODES_TYPE Struct Reference

Contains the topology information for the nodes of a domain.

Collaboration diagram for TYPES::DOMAIN_NODES_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
The pointer to the domain for this nodes topology information.
- INTEGER(INTG) NUMBER_OF_NODES
The number of nodes (excluding ghost nodes) in this domain topology.
- INTEGER(INTG) TOTAL_NUMBER_OF_NODES
The total number of nodes (including ghost nodes) in this domain topology.
- INTEGER(INTG) MAXIMUM_NUMBER_OF_DERIVATIVES
The maximum number of derivatives over the nodes in this domain topology.
- TYPE(DOMAIN_NODE_TYPE), pointer NODES
NODES(np). The pointer to the array of topology information for the nodes of this domain. NODES(np) contains the topological information for the np'th local node of the domain.

7.189.1 Detailed Description

Contains the topology information for the nodes of a domain.

Definition at line 427 of file types.f90.

7.189.2 Member Data Documentation

7.189.2.1 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_NODES_TYPE::DOMAIN

The pointer to the domain for this nodes topology information.

Definition at line 428 of file types.f90.

7.189.2.2 INTEGER(INTG) TYPES::DOMAIN_NODES_TYPE::MAXIMUM_NUMBER_OF_- DERIVATIVES

The maximum number of derivatives over the nodes in this domain topology.

Definition at line 431 of file types.f90.

7.189.2.3 TYPE(DOMAIN_NODE_TYPE),pointer TYPES::DOMAIN_NODES_TYPE::NODES

NODES(np). The pointer to the array of topology information for the nodes of this domain. NODES(np) contains the topological information for the np'th local node of the domain.

Todo

Change this to allocatable???

Definition at line 432 of file types.f90.

7.189.2.4 INTEGER(INTG) TYPES::DOMAIN_NODES_TYPE::NUMBER_OF_NODES

The number of nodes (excluding ghost nodes) in this domain topology.

Definition at line 429 of file types.f90.

7.189.2.5 INTEGER(INTG) TYPES::DOMAIN_NODES_TYPE::TOTAL_NUMBER_OF_NODES

The total number of nodes (including ghost nodes) in this domain topology.

Definition at line 430 of file types.f90.

7.190 TYPES::DOMAIN_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [DOMAIN_TYPE](#).

Collaboration diagram for TYPES::DOMAIN_PTR_TYPE:

Public Attributes

- TYPE([DOMAIN_TYPE](#)), pointer PTR

The pointer to the domain.

7.190.1 Detailed Description

A buffer type to allow for an array of pointers to a [DOMAIN_TYPE](#).

Definition at line 654 of file types.f90.

7.190.2 Member Data Documentation

7.190.2.1 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_PTR_TYPE::PTR

The pointer to the domain.

Definition at line 655 of file types.f90.

7.191 TYPES::DOMAIN_TOPOLOGY_TYPE Struct Reference

Contains the topology information for a domain.

Collaboration diagram for TYPES::DOMAIN_TOPOLOGY_TYPE:

Public Attributes

- TYPE(DOMAIN_TYPE), pointer DOMAIN
The pointer to the domain for this topology information.
- TYPE(DOMAIN_NODES_TYPE), pointer NODES
The pointer to the topology information for the nodes of this domain.
- TYPE(DOMAIN_DOFS_TYPE), pointer DOFS
The pointer to the topology information for the dofs of this domain.
- TYPE(DOMAIN_ELEMENTS_TYPE), pointer ELEMENTS
The pointer to the topology information for the elements of this domain.
- TYPE(DOMAIN_FACES_TYPE), pointer FACES
The pointer to the topology information for the faces of this domain.
- TYPE(DOMAIN_LINES_TYPE), pointer LINES
The pointer to the topology information for the lines of this domain.

7.191.1 Detailed Description

Contains the topology information for a domain.

Definition at line 436 of file types.f90.

7.191.2 Member Data Documentation

7.191.2.1 TYPE(DOMAIN_DOFS_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_- TYPE::DOFS

The pointer to the topology information for the dofs of this domain.

Definition at line 439 of file types.f90.

7.191.2.2 TYPE(DOMAIN_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_TYPE::DOMAIN

The pointer to the domain for this topology information.

Definition at line 437 of file types.f90.

**7.191.2.3 TYPE(DOMAIN_ELEMENTS_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_-
TYPE::ELEMENTS**

The pointer to the topology information for the elements of this domain.

Definition at line 440 of file types.f90.

**7.191.2.4 TYPE(DOMAIN_FACES_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_-
TYPE::FACES**

The pointer to the topology information for the faces of this domain.

Definition at line 441 of file types.f90.

**7.191.2.5 TYPE(DOMAIN_LINES_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_-
TYPE::LINES**

The pointer to the topology information for the lines of this domain.

Definition at line 442 of file types.f90.

**7.191.2.6 TYPE(DOMAIN_NODES_TYPE),pointer TYPES::DOMAIN_TOPOLOGY_-
TYPE::NODES**

The pointer to the topology information for the nodes of this domain.

Definition at line 438 of file types.f90.

7.192 TYPES::DOMAIN_TYPE Struct Reference

A pointer to the domain decomposition for this domain.

Collaboration diagram for TYPES::DOMAIN_TYPE:

Public Attributes

- TYPE(DECOMPOSITION_TYPE), pointer DECOMPOSITION
A pointer to the domain decomposition for this domain.
- TYPE(MESH_TYPE), pointer MESH
A pointer to the mesh for this domain.
- INTEGER(INTG) MESH_COMPONENT_NUMBER
The mesh component number of the mesh which this domain was decomposed from.
- TYPE(REGION_TYPE), pointer REGION
A pointer to the region that this domain is in. This is "inherited" from the mesh region.
- INTEGER(INTG) NUMBER_OF_DIMENSIONS
The number of dimensions for this domain. This is "inherited" from the mesh.
- INTEGER(INTG), allocatable NODE_DOMAIN
*NODE_DOMAIN(np). The domain number that the np'th global node is in for the domain decomposition.
Note: the domain numbers start at 0 and go up to the NUMBER_OF_DOMAINS-1.*
- TYPE(DOMAIN_MAPPINGS_TYPE), pointer MAPPINGS
Pointer to the mappings for the domain e.g., global to local and local to global maps.
- TYPE(DOMAIN_TOPOLOGY_TYPE), pointer TOPOLOGY
Pointer to the topology for the domain.

7.192.1 Detailed Description

A pointer to the domain decomposition for this domain.

Definition at line 642 of file types.f90.

7.192.2 Member Data Documentation

7.192.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::DOMAIN_TYPE::DECOMPOSITION

A pointer to the domain decomposition for this domain.

Definition at line 643 of file types.f90.

7.192.2.2 TYPE(DOMAIN_MAPPINGS_TYPE),pointer TYPES::DOMAIN_TYPE::MAPPINGS

Pointer to the mappings for the domain e.g., global to local and local to global maps.

Definition at line 649 of file types.f90.

7.192.2.3 TYPE(MESH_TYPE),pointer TYPES::DOMAIN_TYPE::MESH

A pointer to the mesh for this domain.

Definition at line 644 of file types.f90.

7.192.2.4 INTEGER(INTG) TYPES::DOMAIN_TYPE::MESH_COMPONENT_NUMBER

The mesh component number of the mesh which this domain was decomposed from.

Definition at line 645 of file types.f90.

7.192.2.5 INTEGER(INTG),allocatable TYPES::DOMAIN_TYPE::NODE_DOMAIN

NODE_DOMAIN(np). The domain number that the np'th global node is in for the domain decomposition.
Note: the domain numbers start at 0 and go up to the NUMBER_OF_DOMAINS-1.

Definition at line 648 of file types.f90.

7.192.2.6 INTEGER(INTG) TYPES::DOMAIN_TYPE::NUMBER_OF_DIMENSIONS

The number of dimensions for this domain. This is "inherited" from the mesh.

Definition at line 647 of file types.f90.

7.192.2.7 TYPE(REGION_TYPE),pointer TYPES::DOMAIN_TYPE::REGION

A pointer to the region that this domain is in. This is "inherited" from the mesh region.

Definition at line 646 of file types.f90.

7.192.2.8 TYPE(DOMAIN_TOPOLOGY_TYPE),pointer TYPES::DOMAIN_TYPE::TOPOLOGY

Pointer to the topology for the domain.

Definition at line 650 of file types.f90.

7.193 TYPES::EIGENPROBLEM_SOLVER_TYPE Struct Reference

Contains information for a time integration solver.

Collaboration diagram for TYPES::EIGENPROBLEM_SOLVER_TYPE:

Public Attributes

- TYPE(SOLVER_TYPE), pointer SOLVER
A pointer to the problem_solver.
- INTEGER(INTG) SOLVER_LIBRARY
The library type for the eigenproblem solver.

7.193.1 Detailed Description

Contains information for a time integration solver.

Definition at line 1282 of file types.f90.

7.193.2 Member Data Documentation

7.193.2.1 TYPE(SOLVER_TYPE),pointer TYPES::EIGENPROBLEM_SOLVER_TYPE::SOLVER

A pointer to the problem_solver.

Definition at line 1283 of file types.f90.

7.193.2.2 INTEGER(INTG) TYPES::EIGENPROBLEM_SOLVER_TYPE::SOLVER_LIBRARY

The library type for the eigenproblem solver.

See also:

[SOLVER_ROUTINES::SolverLibraries](#), [SOLVER_ROUTINES](#)

Definition at line 1284 of file types.f90.

7.194 TYPES::ELEMENT_MATRIX_TYPE Struct Reference

Contains information for an element matrix.

Public Attributes

- INTEGER(INTG) [EQUATIONS_MATRIX_NUMBER](#)
- INTEGER(INTG) [NUMBER_OF_ROWS](#)
- INTEGER(INTG) [NUMBER_OF_COLUMNS](#)
- INTEGER(INTG) [MAX_NUMBER_OF_ROWS](#)
- INTEGER(INTG) [MAX_NUMBER_OF_COLUMNS](#)
- INTEGER(INTG), allocatable [ROW_DOFS](#)
- INTEGER(INTG), allocatable [COLUMN_DOFS](#)
- REAL(DP), allocatable [MATRIX](#)

7.194.1 Detailed Description

Contains information for an element matrix.

Definition at line 937 of file types.f90.

7.194.2 Member Data Documentation

7.194.2.1 INTEGER(INTG),allocatable TYPES::ELEMENT_MATRIX_TYPE::COLUMN_- DOFS

Definition at line 944 of file types.f90.

7.194.2.2 INTEGER(INTG) TYPES::ELEMENT_MATRIX_TYPE::EQUATIONS_MATRIX_- NUMBER

Definition at line 938 of file types.f90.

7.194.2.3 REAL(DP),allocatable TYPES::ELEMENT_MATRIX_TYPE::MATRIX

Definition at line 945 of file types.f90.

7.194.2.4 INTEGER(INTG) TYPES::ELEMENT_MATRIX_TYPE::MAX_NUMBER_OF_- COLUMNS

Definition at line 942 of file types.f90.

7.194.2.5 INTEGER(INTG) TYPES::ELEMENT_MATRIX_TYPE::MAX_NUMBER_OF_- ROWS

Definition at line 941 of file types.f90.

7.194.2.6 INTEGER(INTG) TYPES::ELEMENT_MATRIX_TYPE::NUMBER_OF_COLUMNS

Definition at line 940 of file types.f90.

7.194.2.7 INTEGER(INTG) TYPES::ELEMENT_MATRIX_TYPE::NUMBER_OF_ROWS

Definition at line 939 of file types.f90.

7.194.2.8 INTEGER(INTG),allocatable TYPES::ELEMENT_MATRIX_TYPE::ROW_DOFS

Definition at line 943 of file types.f90.

7.195 TYPES::ELEMENT_VECTOR_TYPE Struct Reference

Contains information for an element vector.

Public Attributes

- INTEGER(INTG) [NUMBER_OF_ROWS](#)
- INTEGER(INTG) [MAX_NUMBER_OF_ROWS](#)
- INTEGER(INTG), allocatable [ROW_DOFS](#)
- REAL(DP), allocatable [VECTOR](#)

7.195.1 Detailed Description

Contains information for an element vector.

Definition at line 949 of file types.f90.

7.195.2 Member Data Documentation

7.195.2.1 INTEGER(INTG) TYPES::ELEMENT_VECTOR_TYPE::MAX_NUMBER_OF_ROWS

Definition at line 951 of file types.f90.

7.195.2.2 INTEGER(INTG) TYPES::ELEMENT_VECTOR_TYPE::NUMBER_OF_ROWS

Definition at line 950 of file types.f90.

7.195.2.3 INTEGER(INTG),allocatable TYPES::ELEMENT_VECTOR_TYPE::ROW_DOFS

Definition at line 952 of file types.f90.

7.195.2.4 REAL(DP),allocatable TYPES::ELEMENT_VECTOR_TYPE::VECTOR

Definition at line 953 of file types.f90.

7.196 TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_- TYPE Struct Reference

Public Attributes

- INTEGER(INTG) [NUMBER_OF_SOLVER_COLS](#)
The number of solver cols this equations column is mapped to.
- INTEGER(INTG), allocatable [SOLVER_COLS](#)
SOLVER_COLS(i). The solver column number for the i'th solver column that this column is mapped to.
- REAL(DP), allocatable [COUPLING_COEFFICIENTS](#)
COUPLING_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.

7.196.1 Detailed Description

Definition at line 1308 of file types.f90.

7.196.2 Member Data Documentation

7.196.2.1 REAL(DP),allocatable TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_- TYPE::COUPLING_COEFFICIENTS

COUPLING_COEFFICIENTS(i). The coupling coefficients for the i'th solver column that this column is mapped to.

Definition at line 1311 of file types.f90.

7.196.2.2 INTEGER(INTG) TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_- TYPE::NUMBER_OF_SOLVER_COLS

The number of solver cols this equations column is mapped to.

Definition at line 1309 of file types.f90.

7.196.2.3 INTEGER(INTG),allocatable TYPES::EQUATIONS_COL_TO_SOLVER_COLS_- MAP_TYPE::SOLVER_COLS

SOLVER_COLS(i). The solver column number for the i'th solver column that this column is mapped to.

Definition at line 1310 of file types.f90.

7.197 TYPES::EQUATIONS_INTERPOLATION_TYPE Struct Reference

Contains information on the interpolation for the equations.

Collaboration diagram for TYPES::EQUATIONS_INTERPOLATION_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS
A pointer to the equations.
- TYPE(FIELD_TYPE), pointer GEOMETRIC_FIELD
A pointer to the geometric field for the equations.
- TYPE(FIELD_TYPE), pointer FIBRE_FIELD
A pointer to the fibre field for the equations (if one is defined).
- TYPE(FIELD_TYPE), pointer DEPENDENT_FIELD
A pointer to the dependent field for the equations.
- TYPE(FIELD_TYPE), pointer MATERIAL_FIELD
A pointer to the material field for the equations (if one is defined).
- TYPE(FIELD_TYPE), pointer SOURCE_FIELD
A pointer to the source field for the equations (if one is defined).
- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer GEOMETRIC_INTERP_PARAMETERS
A pointer to the geometric interpolation parameters for the equations.
- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer FIBRE_INTERP_PARAMETERS
A pointer to the fibre interpolation parameters for the equations (if a fibre field is defined).
- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer DEPENDENT_INTERP_PARAMETERS
A pointer to the dependent interpolation parameters for the equations.
- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer MATERIAL_INTERP_PARAMETERS
A pointer to the material interpolation parameters for the equations (if a material field is defined).
- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer SOURCE_INTERP_PARAMETERS
A pointer to the source interpolation parameters for the equations (if a source field is defined).
- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer GEOMETRIC_INTERP_POINT
A pointer to the geometric interpolated point information for the equations.

- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer FIBRE_INTERP_POINT
A pointer to the fibre interpolated point information for the equations (if a fibre field is defined).
- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer DEPENDENT_INTERP_POINT
A pointer to the dependent interpolated point information for the equations.
- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer MATERIAL_INTERP_POINT
A pointer to the material interpolated point information for the equations (if a material field is defined).
- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer SOURCE_INTERP_POINT
A pointer to the source interpolated point information for the equations (if a source field is defined).
- TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE), pointer GEOMETRIC_INTERP_POINT_METRICS
A pointer to the geometric interpolated point metrics information.
- TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE), pointer FIBRE_INTERP_POINT_METRICS
A pointer to the fibre interpolated point metrics information.

7.197.1 Detailed Description

Contains information on the interpolation for the equations.

Definition at line 1063 of file types.f90.

7.197.2 Member Data Documentation

7.197.2.1 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_TYPE::DEPENDENT_FIELD

A pointer to the dependent field for the equations.

Definition at line 1067 of file types.f90.

7.197.2.2 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_TYPE::DEPENDENT_INTERP_PARAMETERS

A pointer to the dependent interpolation parameters for the equations.

Definition at line 1072 of file types.f90.

7.197.2.3 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_TYPE::DEPENDENT_INTERP_POINT

A pointer to the dependent interpolated point information for the equations.

Definition at line 1077 of file types.f90.

**7.197.2.4 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_-
TYPE::EQUATIONS**

A pointer to the equations.

Definition at line 1064 of file types.f90.

**7.197.2.5 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_-
TYPE::FIBRE_FIELD**

A pointer to the fibre field for the equations (if one is defined).

Definition at line 1066 of file types.f90.

**7.197.2.6 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::FIBRE_INTERP_PARAMETERS**

A pointer to the fibre interpolation parameters for the equations (if a fibre field is defined).

Definition at line 1071 of file types.f90.

**7.197.2.7 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::FIBRE_INTERP_POINT**

A pointer to the fibre interpolated point information for the equations (if a fibre field is defined).

Definition at line 1076 of file types.f90.

**7.197.2.8 TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::FIBRE_INTERP_POINT_-
METRICS**

A pointer to the fibre interpolated point metrics information.

Definition at line 1081 of file types.f90.

**7.197.2.9 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_-
TYPE::GEOMETRIC_FIELD**

A pointer to the geometric field for the equations.

Definition at line 1065 of file types.f90.

**7.197.2.10 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::GEOMETRIC_INTERP_-
PARAMETERS**

A pointer to the geometric interpolation parameters for the equations.

Definition at line 1070 of file types.f90.

**7.197.2.11 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::GEOMETRIC_INTERP_POINT**

A pointer to the geometric interpolated point information for the equations.

Definition at line 1075 of file types.f90.

**7.197.2.12 TYPE(FIELD_INTERPOLATED_POINT_METRICS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::GEOMETRIC_INTERP_-
POINT_METRICS**

A pointer to the geometric interpolated point metrics information.

Definition at line 1080 of file types.f90.

**7.197.2.13 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_-
TYPE::MATERIAL_FIELD**

A pointer to the material field for the equations (if one is defined).

Definition at line 1068 of file types.f90.

**7.197.2.14 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::MATERIAL_INTERP_-
PARAMETERS**

A pointer to the material interpolation parameters for the equations (if a material field is defined).

Definition at line 1073 of file types.f90.

**7.197.2.15 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::MATERIAL_INTERP_POINT**

A pointer to the material interpolated point information for the equations (if a material field is defined).

Definition at line 1078 of file types.f90.

**7.197.2.16 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_INTERPOLATION_-
TYPE::SOURCE_FIELD**

A pointer to the source field for the equations (if one is defined).

Definition at line 1069 of file types.f90.

**7.197.2.17 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::SOURCE_INTERP_-
PARAMETERS**

A pointer to the source interpolation parameters for the equations (if a source field is defined).

Definition at line 1074 of file types.f90.

**7.197.2.18 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer
TYPES::EQUATIONS_INTERPOLATION_TYPE::SOURCE_INTERP_POINT**

A pointer to the source interpolated point information for the equations (if a source field is defined).

Definition at line 1079 of file types.f90.

7.198 TYPES::EQUATIONS_LINEAR_DATA_TYPE Struct Reference

Contains information on any data required for a linear solution.

Collaboration diagram for TYPES::EQUATIONS_LINEAR_DATA_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS
A pointer to the equations.

7.198.1 Detailed Description

Contains information on any data required for a linear solution.

Definition at line 1085 of file types.f90.

7.198.2 Member Data Documentation

7.198.2.1 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_LINEAR_DATA_TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1086 of file types.f90.

7.199 TYPES::EQUATIONS_MAPPING_CREATE_VALUES_- CACHE_TYPE Struct Reference

Public Attributes

- INTEGER(INTG), allocatable [MATRIX_VARIABLE_TYPES](#)

MATRIX_VARIABLE_TYPES(matrix_idx). The dependent variable type mapped to the matrix_idx'th equations matrix.

- REAL(DP), allocatable [MATRIX_COEFFICIENTS](#)

MATRIX_COEFFICIENTS(matrix_idx). The coefficient of the matrix_idx'th matrix in the equations set.

7.199.1 Detailed Description

Definition at line 1030 of file types.f90.

7.199.2 Member Data Documentation

7.199.2.1 REAL(DP),allocatable TYPES::EQUATIONS_MAPPING_CREATE_VALUES_- CACHE_TYPE::MATRIX_COEFFICIENTS

MATRIX_COEFFICIENTS(matrix_idx). The coefficient of the matrix_idx'th matrix in the equations set.

Definition at line 1032 of file types.f90.

7.199.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS_MAPPING_CREATE_- VALUES_CACHE_TYPE::MATRIX_VARIABLE_TYPES

MATRIX_VARIABLE_TYPES(matrix_idx). The dependent variable type mapped to the matrix_idx'th equations matrix.

Definition at line 1031 of file types.f90.

7.200 TYPES::EQUATIONS_MAPPING_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_MAPPING_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS
A pointer to the equations for this equations mapping.
- LOGICAL EQUATIONS_MAPPING_FINISHED
Is .TRUE. if the equations mapping has finished being created, .FALSE. if not.
- TYPE(EQUATIONS_MATRICES_TYPE), pointer EQUATIONS_MATRICES
A pointer to the equations matrices associated with this equations mapping.
- INTEGER(INTG) NUMBER_OF_ROWS
The number of (local) rows in the equations matrices.
- INTEGER(INTG) TOTAL_NUMBER_OF_ROWS
The total number of rows in the equations matrices.
- INTEGER(INTG) NUMBER_OF_EQUATIONS_MATRICES
The number of equations matrices in this mapping.
- INTEGER(INTG) NUMBER_OF_MATRIX_VARIABLES
The number of dependent variables involved in the equations matrix mapping.
- INTEGER(INTG), allocatable MATRIX_VARIABLE_TYPES
MATRIX_VARIABLE_TYPES(i). The variable type of the i'th variable type involved in the equations matrix mapping.
- TYPE(VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE), allocatable VARIABLE_TO_EQUATIONS_MATRICES_MAPS
VARIABLE_TO_EQUATIONS_MATRICES_MAPS(variable_type_idx). The equations matrices mapping for the variable_type_idx'th variable type.
- TYPE(EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE), allocatable EQUATIONS_MATRIX_TO_VARIABLE_MAPS
EQUATIONS_MATRIX_TO_VARIABLE_MAPS(matrix_idx). The mappings for the matrix_idx'th equations matrix.
- TYPE(SOURCE_EQUATIONS_MATRICES_MAP_TYPE), pointer SOURCE_MAPPINGS
A pointer to the mappings between the source and equations matrices if there is a source vector in the equations set.
- TYPE(EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE), allocatable EQUATIONS_ROW_TO_VARIABLES_MAPS
EQUATIONS_ROW_TO_VARIABLES_MAPS(row_idx). The mapping from the row_idx'th row in the equations set to the dependent variables dof.

- INTEGER(INTG) **RHS_VARIABLE_TYPE**
The variable type number mapped to the RHS vector.
- TYPE(**FIELD_VARIABLE_TYPE**), pointer **RHS_VARIABLE**
A pointer to the variable that is mapped to the RHS vector.
- TYPE(**DOMAIN_MAPPING_TYPE**), pointer **RHS_VARIABLE_MAPPING**
A pointer to the RHS variable domain mapping.
- TYPE(**DOMAIN_MAPPING_TYPE**), pointer **ROW_DOFS_MAPPING**
The domain mapping for the equations rows.
- TYPE(**EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE**), pointer **CREATE_VALUES_CACHE**
The create values cache for the equations mapping.

7.200.1 Detailed Description

Definition at line 1035 of file types.f90.

7.200.2 Member Data Documentation

7.200.2.1 TYPE(EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE),pointer TYPES::EQUATIONS_MAPPING_TYPE::CREATE_VALUES_CACHE

The create values cache for the equations mapping.

Definition at line 1053 of file types.f90.

7.200.2.2 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_MAPPING_TYPE::EQUATIONS

A pointer to the equations for this equations mapping.

Definition at line 1036 of file types.f90.

7.200.2.3 LOGICAL TYPES::EQUATIONS_MAPPING_TYPE::EQUATIONS_MAPPING_TYPE::FINISHED

Is .TRUE. if the equations mapping has finished being created, .FALSE. if not.

Definition at line 1037 of file types.f90.

7.200.2.4 TYPE(EQUATIONS_MATRICES_TYPE),pointer TYPES::EQUATIONS_MAPPING_TYPE::EQUATIONS_MATRICES

A pointer to the equations matrices associated with this equations mapping.

Definition at line 1038 of file types.f90.

**7.200.2.5 TYPE(EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE),allocatable
TYPES::EQUATIONS_MAPPING_TYPE::EQUATIONS_MATRIX_TO_VARIABLE_-
MAPS**

EQUATIONS_MATRIX_TO_VARIABLE_MAPS(matrix_idx). The mappings for the matrix_idx'th equations matrix.

Definition at line 1046 of file types.f90.

**7.200.2.6 TYPE(EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE),allocatable
TYPES::EQUATIONS_MAPPING_TYPE::EQUATIONS_ROW_TO_VARIABLES_-
MAPS**

EQUATIONS_ROW_TO_VARIABLES_MAPS(row_idx). The mapping from the row_idx'th row in the equations set to the dependent variables dof.

Definition at line 1048 of file types.f90.

**7.200.2.7 INTEGER(INTG),allocatable TYPES::EQUATIONS_MAPPING_TYPE::MATRIX_-
VARIABLE_TYPES**

MATRIX_VARIABLE_TYPES(i). The variable type of the i'th variable type involved in the equations matrix mapping.

Definition at line 1043 of file types.f90.

**7.200.2.8 INTEGER(INTG) TYPES::EQUATIONS_MAPPING_TYPE::NUMBER_OF_-
EQUATIONS_MATRICES**

The number of equations matrices in this mapping.

Definition at line 1041 of file types.f90.

**7.200.2.9 INTEGER(INTG) TYPES::EQUATIONS_MAPPING_TYPE::NUMBER_OF_-
MATRIX_VARIABLES**

The number of dependent variables involved in the equations matrix mapping.

Definition at line 1042 of file types.f90.

7.200.2.10 INTEGER(INTG) TYPES::EQUATIONS_MAPPING_TYPE::NUMBER_OF_ROWS

The number of (local) rows in the equations matrices.

Definition at line 1039 of file types.f90.

**7.200.2.11 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::EQUATIONS_MAPPING_-
TYPE::RHS_VARIABLE**

A pointer to the variable that is mapped to the RHS vector.

Definition at line 1050 of file types.f90.

**7.200.2.12 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::EQUATIONS_MAPPING_-
TYPE::RHS_VARIABLE_MAPPING**

A pointer to the RHS variable domain mapping.

Definition at line 1051 of file types.f90.

**7.200.2.13 INTEGER(INTG) TYPES::EQUATIONS_MAPPING_TYPE::RHS_VARIABLE_-
TYPE**

The variable type number mapped to the RHS vector.

Definition at line 1049 of file types.f90.

**7.200.2.14 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::EQUATIONS_MAPPING_-
TYPE::ROW_DOFS_MAPPING**

The domain mapping for the equations rows.

Definition at line 1052 of file types.f90.

**7.200.2.15 TYPE(SOURCE_EQUATIONS_MATRICES_MAP_TYPE),pointer
TYPES::EQUATIONS_MAPPING_TYPE::SOURCE_MAPPINGS**

A pointer to the mappings between the source and equations matrices if there is a source vector in the equations set.

Definition at line 1047 of file types.f90.

**7.200.2.16 INTEGER(INTG) TYPES::EQUATIONS_MAPPING_TYPE::TOTAL_NUMBER_-
OF_ROWS**

The total number of rows in the equations matrices.

Definition at line 1040 of file types.f90.

**7.200.2.17 TYPE(VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE),allocatable
TYPES::EQUATIONS_MAPPING_TYPE::VARIABLE_TO_EQUATIONS_-
MATRICES_MAPS**

VARIABLE_TO_EQUATIONS_MATRICES_MAPS(variable_type_idx). The equations matrices mapping for the variable_type_idx'th variable type.

Definition at line 1045 of file types.f90.

7.201 TYPES::EQUATIONS_MATRICES_TYPE Struct Reference

Contains information on the equations matrices and rhs vector.

Collaboration diagram for TYPES::EQUATIONS_MATRICES_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS
A pointer back to the equations.
- LOGICAL EQUATIONS_MATRICES_FINISHED
Is .TRUE. if the equations matrices have finished being created, .FALSE. if not.
- TYPE(EQUATIONS_MAPPING_TYPE), pointer EQUATIONS_MAPPING
A pointer to the equations mapping for the equations matrices.
- TYPE(SOLUTION_MAPPING_TYPE), pointer SOLUTION_MAPPING
A pointer to the solution mapping for the equations matrices.
- INTEGER(INTG) NUMBER_OF_ROWS
The number of rows in the distributed equations matrices.
- INTEGER(INTG) TOTAL_NUMBER_OF_ROWS
The total number of rows in the distributed equations matrices.
- INTEGER(INTG) NUMBER_OF_MATRICES
The number of equations matrices defined for the problem.
- TYPE(EQUATIONS_MATRIX_PTR_TYPE), allocatable MATRICES
MATRICES(matrix_idx) contains the information on the matrix_idx'th equations matrix.
- LOGICAL UPDATE_VECTOR
Is .TRUE. if the equations rhs vector is to be updated.
- TYPE(DISTRIBUTED_VECTOR_TYPE), pointer VECTOR
A pointer to the distributed global rhs vector data.
- TYPE(ELEMENT_VECTOR_TYPE) ELEMENT_VECTOR
The element rhs information.

7.201.1 Detailed Description

Contains information on the equations matrices and rhs vector.

Definition at line 973 of file types.f90.

7.201.2 Member Data Documentation

7.201.2.1 **TYPE(ELEMENT_VECTOR_TYPE) TYPES::EQUATIONS_MATRICES_- TYPE::ELEMENT_VECTOR**

The element rhs information.

Definition at line 984 of file types.f90.

7.201.2.2 **TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_MATRICES_- TYPE::EQUATIONS**

A pointer back to the equations.

Definition at line 974 of file types.f90.

7.201.2.3 **TYPE(EQUATIONS_MAPPING_TYPE),pointer TYPES::EQUATIONS_MATRICES_- TYPE::EQUATIONS_MAPPING**

A pointer to the equations mapping for the equations matrices.

Definition at line 976 of file types.f90.

7.201.2.4 **LOGICAL TYPES::EQUATIONS_MATRICES_TYPE::EQUATIONS_MATRICES_- FINISHED**

Is .TRUE. if the equations matrices have finished being created, .FALSE. if not.

Definition at line 975 of file types.f90.

7.201.2.5 **TYPE(EQUATIONS_MATRIX_PTR_TYPE),allocatable TYPES::EQUATIONS_- MATRICES_TYPE::MATRICES**

MATRICES(matrix_idx) contains the information on the matrix_idx'th equations matrix.

Definition at line 981 of file types.f90.

7.201.2.6 **INTEGER(INTG) TYPES::EQUATIONS_MATRICES_TYPE::NUMBER_OF_- MATRICES**

The number of equations matrices defined for the problem.

Definition at line 980 of file types.f90.

7.201.2.7 **INTEGER(INTG) TYPES::EQUATIONS_MATRICES_TYPE::NUMBER_OF_ROWS**

The number of rows in the distributed equations matrices.

Definition at line 978 of file types.f90.

**7.201.2.8 TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::EQUATIONS_MATRICES_-
TYPE::SOLUTION_MAPPING**

A pointer to the solution mapping for the equations matrices.

Definition at line 977 of file types.f90.

**7.201.2.9 INTEGER(INTG) TYPES::EQUATIONS_MATRICES_TYPE::TOTAL_NUMBER_-
OF_ROWS**

The total number of rows in the distributed equations matrices.

Definition at line 979 of file types.f90.

7.201.2.10 LOGICAL TYPES::EQUATIONS_MATRICES_TYPE::UPDATE_VECTOR

Is .TRUE. if the equations rhs vector is to be updated.

Definition at line 982 of file types.f90.

**7.201.2.11 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::EQUATIONS_-
MATRICES_TYPE::VECTOR**

A pointer to the distributed global rhs vector data.

Definition at line 983 of file types.f90.

7.202 TYPES::EQUATIONS_MATRIX_PTR_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_MATRIX_PTR_TYPE:

Public Attributes

- TYPE(EQUATIONS_MATRIX_TYPE), pointer PTR

7.202.1 Detailed Description

Definition at line 968 of file types.f90.

7.202.2 Member Data Documentation

7.202.2.1 TYPE(EQUATIONS_MATRIX_TYPE),pointer TYPES::EQUATIONS_MATRIX_- PTR_TYPE::PTR

Definition at line 969 of file types.f90.

7.203 TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE:

Public Attributes

- INTEGER(INTG) [MATRIX_NUMBER](#)
The equations matrix number.
- TYPE([EQUATIONS_MATRIX_TYPE](#)), pointer [EQUATIONS_MATRIX](#)
A pointer to the equations matrix.
- INTEGER(INTG) [VARIABLE_TYPE](#)
The dependent variable type mapped to this equations matrix.
- TYPE([FIELD_VARIABLE_TYPE](#)), pointer [VARIABLE](#)
A pointer to the field variable that is mapped to this equations matrix.
- INTEGER(INTG) [NUMBER_OF_COLUMNS](#)
The number of columns in this equations matrix.
- REAL(DP) [MATRIX_COEFFICIENT](#)
The multiplicative coefficient for the matrix in the equation set.
- INTEGER(INTG), allocatable [COLUMN_TO_DOF_MAP](#)
COLUMN_TO_DOF_MAP(column_idx). The variable DOF that the column_idx'th column of this equations matrix is mapped to.
- TYPE([DOMAIN_MAPPING_TYPE](#)), pointer [COLUMN_DOFS_MAPPING](#)
A pointer to the column dofs domain mapping for the matrix variable.

7.203.1 Detailed Description

Definition at line 1014 of file types.f90.

7.203.2 Member Data Documentation

7.203.2.1 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE::COLUMN_DOFS_MAPPING

A pointer to the column dofs domain mapping for the matrix variable.

Definition at line 1022 of file types.f90.

**7.203.2.2 INTEGER(INTG),allocatable TYPES::EQUATIONS_MATRIX_TO_VARIABLE_-
MAP_TYPE::COLUMN_TO_DOF_MAP**

COLUMN_TO_DOF_MAP(column_idx). The variable DOF that the column_idx'th column of this equations matrix is mapped to.

Definition at line 1021 of file types.f90.

**7.203.2.3 TYPE(EQUATIONS_MATRIX_TYPE),pointer TYPES::EQUATIONS_MATRIX_TO_-
VARIABLE_MAP_TYPE::EQUATIONS_MATRIX**

A pointer to the equations matrix.

Definition at line 1016 of file types.f90.

**7.203.2.4 REAL(DP) TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_-
TYPE::MATRIX_COEFFICIENT**

The multiplicative coefficient for the matrix in the equation set.

Definition at line 1020 of file types.f90.

**7.203.2.5 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_-
TYPE::MATRIX_NUMBER**

The equations matrix number.

Definition at line 1015 of file types.f90.

**7.203.2.6 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_-
TYPE::NUMBER_OF_COLUMNS**

The number of columns in this equations matrix.

Definition at line 1019 of file types.f90.

**7.203.2.7 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::EQUATIONS_MATRIX_TO_-
VARIABLE_MAP_TYPE::VARIABLE**

A pointer to the field variable that is mapped to this equations matrix.

Definition at line 1018 of file types.f90.

**7.203.2.8 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_-
TYPE::VARIABLE_TYPE**

The dependent variable type mapped to this equations matrix.

Definition at line 1017 of file types.f90.

7.204 TYPES::EQUATIONS_MATRIX_TYPE Struct Reference

Contains information about an equations matrix.

Collaboration diagram for TYPES::EQUATIONS_MATRIX_TYPE:

Public Attributes

- INTEGER(INTG) [MATRIX_NUMBER](#)
The number of the equations matrix.
- TYPE([EQUATIONS_MATRICES_TYPE](#)), pointer [EQUATIONS_MATRICES](#)
A pointer to the equations matrices for the equation matrix.
- INTEGER(INTG) [STORAGE_TYPE](#)
The storage (sparsity) type for this matrix.
- INTEGER(INTG) [STRUCTURE_TYPE](#)
The structure (sparsity) type for this matrix.
- INTEGER(INTG) [NUMBER_OF_COLUMNS](#)
The number of columns in this global matrix.
- LOGICAL [UPDATE_MATRIX](#)
Is .TRUE. if this global matrix is to be updated.
- TYPE([DISTRIBUTED_MATRIX_TYPE](#)), pointer [MATRIX](#)
A pointer to the distributed global matrix data.
- TYPE([ELEMENT_MATRIX_TYPE](#)) [ELEMENT_MATRIX](#)
The element matrix for this glboal matrix.

7.204.1 Detailed Description

Contains information about an equations matrix.

Definition at line 957 of file types.f90.

7.204.2 Member Data Documentation

7.204.2.1 TYPE(ELEMENT_MATRIX_TYPE) TYPES::EQUATIONS_MATRIX_- TYPE::ELEMENT_MATRIX

The element matrix for this glboal matrix.

Definition at line 965 of file types.f90.

**7.204.2.2 TYPE(EQUATIONS_MATRICES_TYPE),pointer TYPES::EQUATIONS_MATRIX_-
TYPE::EQUATIONS_MATRICES**

A pointer to the equations matrices for the equation matrix.

Definition at line 959 of file types.f90.

**7.204.2.3 TYPE(DISTRIBUTED_MATRIX_TYPE),pointer TYPES::EQUATIONS_MATRIX_-
TYPE::MATRIX**

A pointer to the distributed global matrix data.

Definition at line 964 of file types.f90.

7.204.2.4 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TYPE::MATRIX_NUMBER

The number of the equations matrix.

Definition at line 958 of file types.f90.

**7.204.2.5 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TYPE::NUMBER_OF_-
COLUMNS**

The number of columns in this global matrix.

Definition at line 962 of file types.f90.

7.204.2.6 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TYPE::STORAGE_TYPE

The storage (sparsity) type for this matrix.

Definition at line 960 of file types.f90.

7.204.2.7 INTEGER(INTG) TYPES::EQUATIONS_MATRIX_TYPE::STRUCTURE_TYPE

The structure (sparsity) type for this matrix.

Definition at line 961 of file types.f90.

7.204.2.8 LOGICAL TYPES::EQUATIONS_MATRIX_TYPE::UPDATE_MATRIX

Is .TRUE. if this global matrix is to be updated.

Definition at line 963 of file types.f90.

7.205 TYPES::EQUATIONS_NONLINEAR_DATA_TYPE Struct Reference

Contains information on any data required for a non-linear solution.

Collaboration diagram for TYPES::EQUATIONS_NONLINEAR_DATA_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS

A pointer to the equations.

7.205.1 Detailed Description

Contains information on any data required for a non-linear solution.

Definition at line 1090 of file types.f90.

7.205.2 Member Data Documentation

7.205.2.1 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_NONLINEAR_DATA_TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1091 of file types.f90.

7.206 TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_- TYPE Struct Reference

Contains information on the mapping from the equations rows in an equations set to the solver rows.

Public Attributes

- INTEGER(INTG) **NUMBER_OF_SOLVER_ROWS**
The number of solver rows this equations row is mapped to.
- INTEGER(INTG), allocatable **SOLVER_ROWS**
SOLVER_ROWS(i). The solver row number for the i'th solver row that this row is mapped to.
- REAL(DP), allocatable **COUPLING_COEFFICIENTS**
COUPLING_COEFFICIENTS(i). The coupling coefficients for the i'th solver row that this row is mapped to.

7.206.1 Detailed Description

Contains information on the mapping from the equations rows in an equations set to the solver rows.

Definition at line 1354 of file types.f90.

7.206.2 Member Data Documentation

7.206.2.1 REAL(DP),allocatable TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_- TYPE::COUPLING_COEFFICIENTS

COUPLING_COEFFICIENTS(i). The coupling coefficients for the i'th solver row that this row is mapped to.

Definition at line 1357 of file types.f90.

7.206.2.2 INTEGER(INTG) TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_- TYPE::NUMBER_OF_SOLVER_ROWS

The number of solver rows this equations row is mapped to.

Definition at line 1355 of file types.f90.

7.206.2.3 INTEGER(INTG),allocatable TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_- MAP_TYPE::SOLVER_ROWS

SOLVER_ROWS(i). The solver row number for the i'th solver row that this row is mapped to.

Definition at line 1356 of file types.f90.

7.207 TYPES::EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE Struct Reference

Public Attributes

- INTEGER(INTG), allocatable [ROW_TO_DOFS_MAP](#)

ROW_TO_DOFS_MAP(i). The variable dof number for the i'th variable that this equations row is mapped to.

- INTEGER(INTG) [ROW_TO_RHS_DOF](#)

The RHS variable dof number that this row is mapped to. If there is no RHS vector in this equations set the variable dof number is zero.

7.207.1 Detailed Description

Definition at line 1025 of file types.f90.

7.207.2 Member Data Documentation

7.207.2.1 INTEGER(INTG),allocatable TYPES::EQUATIONS_ROW_TO_VARIABLE_MAP_- TYPE::ROW_TO_DOFS_MAP

ROW_TO_DOFS_MAP(i). The variable dof number for the i'th variable that this equations row is mapped to.

Definition at line 1026 of file types.f90.

7.207.2.2 INTEGER(INTG) TYPES::EQUATIONS_ROW_TO_VARIABLE_MAP_- TYPE::ROW_TO_RHS_DOF

The RHS variable dof number that this row is mapped to. If there is no RHS vector in this equations set the variable dof number is zero.

Definition at line 1027 of file types.f90.

7.208 TYPES::EQUATIONS_SET_ANALYTIC_TYPE Struct Reference

Contains information on the analytic setup for the equations set.

Collaboration diagram for TYPES::EQUATIONS_SET_ANALYTIC_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- LOGICAL ANALYTIC_FINISHED
Is .TRUE. if the analytic setup for the problem has finished being created, .FALSE. if not.

7.208.1 Detailed Description

Contains information on the analytic setup for the equations set.

Definition at line 1157 of file types.f90.

7.208.2 Member Data Documentation

7.208.2.1 LOGICAL TYPES::EQUATIONS_SET_ANALYTIC_TYPE::ANALYTIC_FINISHED

Is .TRUE. if the analytic setup for the problem has finished being created, .FALSE. if not.

Definition at line 1159 of file types.f90.

7.208.2.2 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_ANALYTIC_TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1158 of file types.f90.

7.209 TYPES::EQUATIONS_SET_DEPENDENT_TYPE Struct Reference

Contains information on the dependent variables for the equations set.

Collaboration diagram for TYPES::EQUATIONS_SET_DEPENDENT_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- LOGICAL DEPENDENT_FINISHED
Is .TRUE. if the dependent variables for the equations set has finished being created, .FALSE. if not.
- TYPE(FIELD_TYPE), pointer DEPENDENT_FIELD
A pointer to the dependent field for the equations set.

7.209.1 Detailed Description

Contains information on the dependent variables for the equations set.

Definition at line 1143 of file types.f90.

7.209.2 Member Data Documentation

7.209.2.1 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_SET_DEPENDENT_- TYPE::DEPENDENT_FIELD

A pointer to the dependent field for the equations set.

Definition at line 1146 of file types.f90.

7.209.2.2 LOGICAL TYPES::EQUATIONS_SET_DEPENDENT_TYPE::DEPENDENT_- FINISHED

Is .TRUE. if the dependent variables for the equations set has finished being created, .FALSE. if not.

Definition at line 1145 of file types.f90.

7.209.2.3 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_- DEPENDENT_TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1144 of file types.f90.

7.210 TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE Struct Reference

Contains information on the fixed conditions for the problem.

Collaboration diagram for TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- LOGICAL FIXED_CONDITIONS_FINISHED
Is .TRUE. if the fixed conditions for the equations set has finished being created, .FALSE. if not.
- INTEGER(INTG), allocatable GLOBAL_BOUNDARY_CONDITIONS
GLOBAL_BOUNDARY_CONDITIONS(dof_idx). The global boundary condition for the dof_idx'th dof of the dependent field.
- TYPE(DISTRIBUTED_VECTOR_TYPE), pointer BOUNDARY_CONDITIONS
A pointer to the distributed vector containing the boundary conditions for the domain for this process.

7.210.1 Detailed Description

Contains information on the fixed conditions for the problem.

Definition at line 1135 of file types.f90.

7.210.2 Member Data Documentation

7.210.2.1 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE::BOUNDARY_CONDITIONS

A pointer to the distributed vector containing the boundary conditions for the domain for this process.

Definition at line 1139 of file types.f90.

7.210.2.2 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1136 of file types.f90.

7.210.2.3 LOGICAL TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE::FIXED_CONDITIONS_FINISHED

Is .TRUE. if the fixed conditions for the equations set has finished being created, .FALSE. if not.

Definition at line 1137 of file types.f90.

**7.210.2.4 INTEGER(INTG),allocatable TYPES::EQUATIONS_SET_FIXED_CONDITIONS_-
TYPE::GLOBAL_BOUNDARY_CONDITIONS**

GLOBAL_BOUNDARY_CONDITIONS(dof_idx). The global boundary condition for the dof_idx'th dof of the dependent field.

See also:

EQUATIONS_ROUTINES_FixedConditions,EQUATIONS_ROUTINES

Definition at line 1138 of file types.f90.

7.211 TYPES::EQUATIONS_SET_GEOMETRY_TYPE Struct Reference

Contains information on the geometry for an equations set.

Collaboration diagram for TYPES::EQUATIONS_SET_GEOMETRY_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- TYPE(FIELD_TYPE), pointer GEOMETRIC_FIELD
The geometric field for this equations set.
- TYPE(FIELD_TYPE), pointer FIBRE_FIELD
The fibre field for this equations set if one is defined. If no fibre field is defined the pointer is NULL.

7.211.1 Detailed Description

Contains information on the geometry for an equations set.

Definition at line 1122 of file types.f90.

7.211.2 Member Data Documentation

7.211.2.1 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_GEOMETRY_- TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1123 of file types.f90.

7.211.2.2 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_SET_GEOMETRY_- TYPE::FIBRE_FIELD

The fibre field for this equations set if one is defined. If no fibre field is defined the pointer is NULL.

Definition at line 1125 of file types.f90.

7.211.2.3 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_SET_GEOMETRY_- TYPE::GEOMETRIC_FIELD

The geometric field for this equations set.

Definition at line 1124 of file types.f90.

7.212 TYPES::EQUATIONS_SET_MATERIALS_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_SET_MATERIALS_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- LOGICAL MATERIALS_FINISHED
Is .TRUE. if the materials for the equations set has finished being created, .FALSE. if not.
- TYPE(FIELD_TYPE), pointer MATERIAL_FIELD
A pointer to the material field for the equations set if one is defined. If no material field is defined the pointer is NULL.

7.212.1 Detailed Description

Definition at line 1128 of file types.f90.

7.212.2 Member Data Documentation

7.212.2.1 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_- MATERIALS_TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1129 of file types.f90.

7.212.2.2 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_SET_MATERIALS_- TYPE::MATERIAL_FIELD

A pointer to the material field for the equations set if one is defined. If no material field is defined the pointer is NULL.

Definition at line 1131 of file types.f90.

7.212.2.3 LOGICAL TYPES::EQUATIONS_SET_MATERIALS_TYPE::MATERIALS_- FINISHED

Is .TRUE. if the materials for the equations set has finished being created, .FALSE. if not.

Definition at line 1130 of file types.f90.

7.213 TYPES::EQUATIONS_SET_PTR_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_SET_PTR_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer PTR

7.213.1 Detailed Description

Definition at line 1189 of file types.f90.

7.213.2 Member Data Documentation

7.213.2.1 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_PTR_- TYPE::PTR

Definition at line 1190 of file types.f90.

7.214 TYPES::EQUATIONS_SET_SOURCE_TYPE Struct Reference

Contains information on the source for the equations set.

Collaboration diagram for TYPES::EQUATIONS_SET_SOURCE_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations set.
- LOGICAL SOURCE_FINISHED
Is .TRUE. if the source for the equations set has finished being created, .FALSE. if not.
- TYPE(FIELD_TYPE), pointer SOURCE_FIELD
A pointer to the source field for the equations set if one is defined. If no source is defined the pointer is NULL.

7.214.1 Detailed Description

Contains information on the source for the equations set.

Definition at line 1150 of file types.f90.

7.214.2 Member Data Documentation

7.214.2.1 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_SET_SOURCE_- TYPE::EQUATIONS_SET

A pointer to the equations set.

Definition at line 1151 of file types.f90.

7.214.2.2 TYPE(FIELD_TYPE),pointer TYPES::EQUATIONS_SET_SOURCE_- TYPE::SOURCE_FIELD

A pointer to the source field for the equations set if one is defined. If no source is defined the pointer is NULL.

Definition at line 1153 of file types.f90.

7.214.2.3 LOGICAL TYPES::EQUATIONS_SET_SOURCE_TYPE::SOURCE_FINISHED

Is .TRUE. if the source for the equations set has finished being created, .FALSE. if not.

Definition at line 1152 of file types.f90.

7.215 TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE Struct Reference

Contains information on the mappings from an equations set to the solver matrices.

Collaboration diagram for TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE:

Public Attributes

- INTEGER(INTG) [EQUATIONS_SET_INDEX](#)
The index of the equations set for these mappings.
- TYPE([SOLUTION_MAPPING_TYPE](#)), pointer [SOLUTION_MAPPING](#)
A pointer to the solution mapping.
- TYPE([EQUATIONS_TYPE](#)), pointer [EQUATIONS](#)
A pointer to the equations in this equations set.
- TYPE([EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE](#)), allocatable [EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM](#)
EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM(solver_matrix_idx). The mappings from the equations matrices in this equation set to the solver_matrix_idx'th solver_matrix.
- TYPE([EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE](#)), allocatable [EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM](#)
EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM(equations_matrix_idx). The mappings from the equation_matrix_idx'th equations matrix in this equation set to the solver_matrices.
- TYPE([EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE](#)), allocatable [EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS](#)
EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS(equations_row_idx). The mappings from the equations_row_idx'th equations row to the solver matrices rows.

7.215.1 Detailed Description

Contains information on the mappings from an equations set to the solver matrices.

Definition at line 1361 of file types.f90.

7.215.2 Member Data Documentation

7.215.2.1 TYPE([EQUATIONS_TYPE](#)),pointer TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE::[EQUATIONS](#)

A pointer to the equations in this equations set.

Definition at line 1364 of file types.f90.

**7.215.2.2 TYPE(EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE),allocatable
TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE::EQUATIONS_ROW_TO_-
SOLVER_ROWS_MAPS**

EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS(equations_row_idx). The mappings from the equations_row_idx'th equations row to the solver matrices rows.

Definition at line 1367 of file types.f90.

**7.215.2.3 INTEGER(INTG) TYPES::EQUATIONS_SET_TO_SOLVER_MAP_-
TYPE::EQUATIONS_SET_INDEX**

The index of the equations set for these mappings.

Definition at line 1362 of file types.f90.

**7.215.2.4 TYPE(EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE),allocatable
TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE::EQUATIONS_TO_-
SOLVER_MATRIX_MAPS_EM**

EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM(equations_matrix_idx). The mappings from the equation_matrix_idx'th equations matrix in this equation set to the solver_matrices.

Definition at line 1366 of file types.f90.

**7.215.2.5 TYPE(EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE),allocatable
TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE::EQUATIONS_TO_-
SOLVER_MATRIX_MAPS_SM**

EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM(solver_matrix_idx). The mappings from the equations matrices in this equation set to the solver_matrix_idx'th solver_matrix.

Definition at line 1365 of file types.f90.

**7.215.2.6 TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::EQUATIONS_SET_TO_-
SOLVER_MAP_TYPE::SOLUTION_MAPPING**

A pointer to the solution mappings.

Definition at line 1363 of file types.f90.

7.216 TYPES::EQUATIONS_SET_TYPE Struct Reference

Contains information on an equations set.

Collaboration diagram for TYPES::EQUATIONS_SET_TYPE:

Public Attributes

- INTEGER(INTG) **USER_NUMBER**
The user identifying number of the equations set.
- INTEGER(INTG) **GLOBAL_NUMBER**
The global index of the equations set in the region.
- LOGICAL **EQUATIONS_SET_FINISHED**
Is .TRUE. if the equations set have finished being created, .FALSE. if not.
- TYPE(EQUATIONS_SETS_TYPE), pointer **EQUATIONS_SETS**
A pointer back to the equations sets.
- TYPE(REGION_TYPE), pointer **REGION**
A pointer back to the region containing the equations set.
- INTEGER(INTG) **CLASS**
The equations set specification class identifier.
- INTEGER(INTG) **TYPE**
The equations set specification type identifier.
- INTEGER(INTG) **SUBTYPE**
The equations set specification subtype identifier.
- INTEGER(INTG) **LINEARITY**
The equations set linearity type.
- INTEGER(INTG) **TIME_TYPE**
The equations set time dependence type.
- INTEGER(INTG) **SOLUTION_METHOD**
The solution method for the equations set.
- TYPE(EQUATIONS_SET_GEOMETRY_TYPE) **GEOMETRY**
The geometry information for the equations set.
- TYPE(EQUATIONS_SET_MATERIALS_TYPE), pointer **MATERIALS**
A pointer to the materials information for the equations set.
- TYPE(EQUATIONS_SET_SOURCE_TYPE), pointer **SOURCE**
A pointer to the source information for the equations set.

- **TYPE(EQUATIONS_SET_DEPENDENT_TYPE) DEPENDENT**

The depedent variable information for the equations set.

- **TYPE(EQUATIONS_SET_ANALYTIC_TYPE), pointer ANALYTIC**

A pointer to the analytic setup information for the equations set.

- **TYPE(EQUATIONS_SET_FIXED_CONDITIONS_TYPE), pointer FIXED_CONDITIONS**

A pointer to the fixed condition information for the equations set.

- **TYPE(EQUATIONS_TYPE), pointer EQUATIONS**

7.216.1 Detailed Description

Contains information on an equations set.

Definition at line 1163 of file types.f90.

7.216.2 Member Data Documentation

7.216.2.1 TYPE(EQUATIONS_SET_ANALYTIC_TYPE),pointer TYPES::EQUATIONS_SET_- TYPE::ANALYTIC

A pointer to the analytic setup information for the equations set.

Definition at line 1184 of file types.f90.

7.216.2.2 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::CLASS

The equations set specification class identifier.

Definition at line 1170 of file types.f90.

7.216.2.3 TYPE(EQUATIONS_SET_DEPENDENT_TYPE) TYPES::EQUATIONS_SET_- TYPE::DEPENDENT

The depedent variable information for the equations set.

Definition at line 1183 of file types.f90.

7.216.2.4 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_SET_TYPE::EQUATIONS

Definition at line 1186 of file types.f90.

7.216.2.5 LOGICAL TYPES::EQUATIONS_SET_TYPE::EQUATIONS_SET_FINISHED

Is .TRUE. if the equations set have finished being created, .FALSE. if not.

Definition at line 1166 of file types.f90.

**7.216.2.6 TYPE(EQUATIONS_SETS_TYPE),pointer TYPES::EQUATIONS_SET_-
TYPE::EQUATIONS_SETS**

A pointer back to the equations sets.

Definition at line 1167 of file types.f90.

**7.216.2.7 TYPE(EQUATIONS_SET_FIXED_CONDITIONS_TYPE),pointer
TYPES::EQUATIONS_SET_TYPE::FIXED_CONDITIONS**

A pointer to the fixed condition information for the equations set.

Todo

Change name to BOUNDARY_CONDITIONS???

Definition at line 1185 of file types.f90.

**7.216.2.8 TYPE(EQUATIONS_SET_GEOMETRY_TYPE) TYPES::EQUATIONS_SET_-
TYPE::GEOMETRY**

The geometry information for the equations set.

Definition at line 1180 of file types.f90.

7.216.2.9 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::GLOBAL_NUMBER

The global index of the equations set in the region.

Definition at line 1165 of file types.f90.

7.216.2.10 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::LINEARITY

The equations set linearity type.

See also:

EQUATIONS_ROUTINES_LinearityTypes

Definition at line 1175 of file types.f90.

**7.216.2.11 TYPE(EQUATIONS_SET_MATERIALS_TYPE),pointer
TYPES::EQUATIONS_SET_TYPE::MATERIALS**

A pointer to the materials information for the equations set.

Definition at line 1181 of file types.f90.

7.216.2.12 TYPE(REGION_TYPE),pointer TYPES::EQUATIONS_SET_TYPE::REGION

A pointer back to the region containing the equations set.

Definition at line 1168 of file types.f90.

7.216.2.13 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::SOLUTION_METHOD

The solution method for the equations set.

See also:

EQUATIONS_ROUTINES_SolutionMethods

Definition at line 1178 of file types.f90.

7.216.2.14 TYPE(EQUATIONS_SET_SOURCE_TYPE),pointer TYPES::EQUATIONS_SET_TYPE::SOURCE

A pointer to the source information for the equations set.

Definition at line 1182 of file types.f90.

7.216.2.15 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::SUBTYPE

The equations set specification subtype identifier.

Definition at line 1172 of file types.f90.

7.216.2.16 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::TIME_TYPE

The equations set time dependence type.

See also:

EQUATIONS_ROUTINES_TimeDepedenceTypes

Definition at line 1176 of file types.f90.

7.216.2.17 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::TYPE

The equations set specification type identifier.

Definition at line 1171 of file types.f90.

7.216.2.18 INTEGER(INTG) TYPES::EQUATIONS_SET_TYPE::USER_NUMBER

The user identifying number of the equations set.

Definition at line 1164 of file types.f90.

7.217 TYPES::EQUATIONS_SETS_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_SETS_TYPE:

Public Attributes

- TYPE(REGION_TYPE), pointer REGION
- INTEGER(INTG) NUMBER_OF_EQUATIONS_SETS
- TYPE(EQUATIONS_SET_PTR_TYPE), pointer EQUATIONS_SETS

7.217.1 Detailed Description

Definition at line 1193 of file types.f90.

7.217.2 Member Data Documentation

7.217.2.1 TYPE(EQUATIONS_SET_PTR_TYPE),pointer TYPES::EQUATIONS_SETS_- TYPE::EQUATIONS_SETS

Definition at line 1196 of file types.f90.

7.217.2.2 INTEGER(INTG) TYPES::EQUATIONS_SETS_TYPE::NUMBER_OF_- EQUATIONS_SETS

Definition at line 1195 of file types.f90.

7.217.2.3 TYPE(REGION_TYPE),pointer TYPES::EQUATIONS_SETS_TYPE::REGION

Definition at line 1194 of file types.f90.

7.218 TYPES::EQUATIONS_TIME_DATA_TYPE Struct Reference

Contains information on any data required for a time-dependent equations.

Collaboration diagram for TYPES::EQUATIONS_TIME_DATA_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS

A pointer to the equations.

7.218.1 Detailed Description

Contains information on any data required for a time-dependent equations.

Definition at line 1095 of file types.f90.

7.218.2 Member Data Documentation

7.218.2.1 TYPE(EQUATIONS_TYPE),pointer TYPES::EQUATIONS_TIME_DATA_TYPE::EQUATIONS

A pointer to the equations.

Definition at line 1096 of file types.f90.

7.219 TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a EQUATIONS_TO_SOLVER_MAPS_TYPE.

Collaboration diagram for TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE:

Public Attributes

- TYPE(EQUATIONS_TO_SOLVER_MAPS_TYPE), pointer PTR
A pointer to the equations to solver maps.

7.219.1 Detailed Description

A buffer type to allow for an array of pointers to a EQUATIONS_TO_SOLVER_MAPS_TYPE.

See also:

TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE

Definition at line 1323 of file types.f90.

7.219.2 Member Data Documentation

7.219.2.1 TYPE(EQUATIONS_TO_SOLVER_MAPS_TYPE),pointer TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE::PTR

A pointer to the equations to solver maps.

Definition at line 1324 of file types.f90.

7.220 TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE Struct Reference

Collaboration diagram for TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE:

Public Attributes

- INTEGER(INTG) [EQUATIONS_MATRIX_NUMBER](#)
The equations matrix number being mapped.
- INTEGER(INTG) [SOLVER_MATRIX_NUMBER](#)
The solver matrix number being mapped.
- TYPE([EQUATIONS_MATRIX_TYPE](#)), pointer [EQUATIONS_MATRIX](#)
A pointer to the equations matrix being mapped.
- TYPE([SOLVER_MATRIX_TYPE](#)), pointer [SOLVER_MATRIX](#)
A pointer to the solver matrix being mapped.
- TYPE([EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE](#)), allocatable [EQUATIONS_COL_-
SOLVER_COLS_MAP](#)
*EQUATIONS_COL_SOLVER_COL_MAP(column_idx). The mapping from the column_idx'th column of
this equations matrix to the solver matrix columns.*

7.220.1 Detailed Description

Definition at line 1314 of file types.f90.

7.220.2 Member Data Documentation

7.220.2.1 TYPE([EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE](#)),allocatable TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE::EQUATIONS_COL_SOLVER_- COLS_MAP

EQUATIONS_COL_SOLVER_COL_MAP(column_idx). The mapping from the column_idx'th column of this equations matrix to the solver matrix columns.

Definition at line 1319 of file types.f90.

7.220.2.2 TYPE([EQUATIONS_MATRIX_TYPE](#)),pointer TYPES::EQUATIONS_TO_SOLVER_- MAPS_TYPE::EQUATIONS_MATRIX

A pointer to the equations matrix being mapped.

Definition at line 1317 of file types.f90.

**7.220.2.3 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MAPS_-
TYPE::EQUATIONS_MATRIX_NUMBER**

The equations matrix number being mapped.

Definition at line 1315 of file types.f90.

**7.220.2.4 TYPE(SOLVER_MATRIX_TYPE),pointer TYPES::EQUATIONS_TO_SOLVER_-
MAPS_TYPE::SOLVER_MATRIX**

A pointer to the solver matrix being mapped.

Definition at line 1318 of file types.f90.

**7.220.2.5 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE::SOLVER_-
MATRIX_NUMBER**

The solver matrix number being mapped.

Definition at line 1316 of file types.f90.

7.221 TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE Struct Reference

Contains information on the equations to solver matrix mappings when indexing by equations matrix number.

Collaboration diagram for TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE:

Public Attributes

- INTEGER(INTG) [EQUATIONS_MATRIX_NUMBER](#)
The number of the equations matrix for these mappings.
- INTEGER(INTG) [NUMBER_OF_SOLVER_MATRICES](#)
The number of solver matrices mapped to this equations matrix.
- TYPE([EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE](#)), allocatable [EQUATIONS_TO_SOLVER_MATRIX_MAPS](#)
EQUATIONS_TO_SOLVER_MATRIX_MAPS(solver_matrix_idx). The maps from the equation matrix to the solver_matrix_idx'th solver matrix.

7.221.1 Detailed Description

Contains information on the equations to solver matrix mappings when indexing by equations matrix number.

Definition at line 1347 of file types.f90.

7.221.2 Member Data Documentation

7.221.2.1 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE::EQUATIONS_MATRIX_NUMBER

The number of the equations matrix for these mappings.

Definition at line 1348 of file types.f90.

7.221.2.2 TYPE(EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE),allocatable TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE::EQUATIONS_TO_SOLVER_MATRIX_MAPS

EQUATIONS_TO_SOLVER_MATRIX_MAPS(solver_matrix_idx). The maps from the equation matrix to the solver_matrix_idx'th solver matrix.

Definition at line 1350 of file types.f90.

7.221.2.3 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE::NUMBER_OF_SOLVER_MATRICES

The number of solver matrices mapped to this equations matrix.

Definition at line 1349 of file types.f90.

7.222 TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE Struct Reference

Contains information on the equations to solver matrix mappings when indexing by solver matrix number.

Collaboration diagram for TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE:

Public Attributes

- INTEGER(INTG) **SOLVER_MATRIX_NUMBER**
The number of the solver matrix for these mappings.
- INTEGER(INTG) **NUMBER_OF_VARIABLES**
The number of variables mapped to this solver matrix.
- INTEGER(INTG), allocatable **VARIABLE_TYPES**
VARIABLE_TYPES(variable_idx). The variable type for the variable_idx'th variable that is mapped to the solver matrix.
- TYPE(**FIELD_VARIABLE_PTR_TYPE**), allocatable **VARIABLES**
VARIABLES(variable_idx). VARIABLES(variable_idx)PTR points to the variable_idx'th variable that is mapped to the solver matrix.
- TYPE(**VARIABLE_TO_SOLVER_COL_MAP_TYPE**), allocatable **VARIABLE_TO_SOLVER_COL_MAPS**
VARIABLE_TO_SOLVER_COL_MAPS(variable_idx). The mappings from the variable dofs to the solver dofs for the variable_idx'th variable in the equations set that is mapped to the solver matrix.
- INTEGER(INTG) **NUMBER_OF_EQUATIONS_MATRICES**
The number of equations matrices mapped to this solver matrix.
- TYPE(**EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE**), allocatable **EQUATIONS_TO_SOLVER_MATRIX_MAPS**
EQUATIONS_TO_SOLVER_MATRIX_MAPS(equations_matrix_idx). The maps from the equations_idx'th equations matrix to solver matrix.

7.222.1 Detailed Description

Contains information on the equations to solver matrix mappings when indexing by solver matrix number.
 Definition at line 1335 of file types.f90.

7.222.2 Member Data Documentation

7.222.2.1 TYPE(EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE),allocatable TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE::EQUATIONS_TO_SOLVER_MATRIX_MAPS

EQUATIONS_TO_SOLVER_MATRIX_MAPS(equations_matrix_idx). The maps from the equations_idx'th equations matrix to solver matrix.

Definition at line 1343 of file types.f90.

**7.222.2.2 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_-
TYPE::NUMBER_OF_EQUATIONS_MATRICES**

The number of equations matrices mapped to this solver matrix.

Definition at line 1342 of file types.f90.

**7.222.2.3 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_-
TYPE::NUMBER_OF_VARIABLES**

The number of variables mapped to this solver matrix.

Definition at line 1338 of file types.f90.

**7.222.2.4 INTEGER(INTG) TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_-
TYPE::SOLVER_MATRIX_NUMBER**

The number of the solver matrix for these mappings.

Definition at line 1336 of file types.f90.

**7.222.2.5 TYPE(VARIABLE_TO_SOLVER_COL_MAP_TYPE),allocatable
TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE::VARIABLE_-
TO_SOLVER_COL_MAPS**

VARIABLE_TO_SOLVER_COL_MAPS(variable_idx). The mappings from the variable dofs to the solver dofs for the variable_idx'th variable in the equations set that is mapped to the solver matrix.

Definition at line 1341 of file types.f90.

**7.222.2.6 INTEGER(INTG),allocatable TYPES::EQUATIONS_TO_SOLVER_MATRIX_-
MAPS_SM_TYPE::VARIABLE_TYPES**

VARIABLE_TYPES(variable_idx). The variable type for the variable_idx'th variable that is mapped to the solver matrix.

Definition at line 1339 of file types.f90.

**7.222.2.7 TYPE(FIELD_VARIABLE_PTR_TYPE),allocatable TYPES::EQUATIONS_TO_-
SOLVER_MATRIX_MAPS_SM_TYPE::VARIABLES**

VARIABLES(variable_idx). VARIABLES(variable_idx)PTR points to the variable_idx'th variable that is mapped to the solver matrix.

Definition at line 1340 of file types.f90.

7.223 TYPES::EQUATIONS_TYPE Struct Reference

Contains information about the equations in an equations set.

Collaboration diagram for TYPES::EQUATIONS_TYPE:

Public Attributes

- TYPE(EQUATIONS_SET_TYPE), pointer EQUATIONS_SET
A pointer to the equations_set.
- LOGICAL EQUATIONS_FINISHED
Is .TRUE. if the equations have finished being created, .FALSE. if not.
- INTEGER(INTG) OUTPUT_TYPE
The output type for the equations.
- INTEGER(INTG) SPARSITY_TYPE
The sparsity type for the equation matrices of the equations.
- TYPE(EQUATIONS_INTERPOLATION_TYPE), pointer INTERPOLATION
A pointer to the interpolation information used in the equations.
- TYPE(EQUATIONS_LINEAR_DATA_TYPE), pointer LINEAR_DATA
A pointer to the data for linear equations.
- TYPE(EQUATIONS_NONLINEAR_DATA_TYPE), pointer NONLINEAR_DATA
A pointer to the data for non-linear equations.
- TYPE(EQUATIONS_TIME_DATA_TYPE), pointer TIME_DATA
A pointer to the data for non-static equations.
- TYPE(EQUATIONS_MAPPING_TYPE), pointer EQUATIONS_MAPPING
A pointer to the equations mapping for the equations.
- TYPE(EQUATIONS_MATRICES_TYPE), pointer EQUATIONS_MATRICES
A pointer to the equations matrices and vectors used for the equations.

7.223.1 Detailed Description

Contains information about the equations in an equations set.

Definition at line 1100 of file types.f90.

7.223.2 Member Data Documentation

7.223.2.1 LOGICAL TYPES::EQUATIONS_TYPE::EQUATIONS_FINISHED

Is .TRUE. if the equations have finished being created, .FALSE. if not.

Definition at line 1102 of file types.f90.

**7.223.2.2 TYPE(EQUATIONS_MAPPING_TYPE),pointer TYPES::EQUATIONS_-
TYPE::EQUATIONS_MAPPING**

A pointer to the equations mapping for the equations.

Definition at line 1111 of file types.f90.

**7.223.2.3 TYPE(EQUATIONS_MATRICES_TYPE),pointer TYPES::EQUATIONS_-
TYPE::EQUATIONS_MATRICES**

A pointer to the equations matrices and vectors used for the equations.

Definition at line 1112 of file types.f90.

**7.223.2.4 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::EQUATIONS_-
TYPE::EQUATIONS_SET**

A pointer to the equations_set.

Definition at line 1101 of file types.f90.

**7.223.2.5 TYPE(EQUATIONS_INTERPOLATION_TYPE),pointer TYPES::EQUATIONS_-
TYPE::INTERPOLATION**

A pointer to the interpolation information used in the equations.

Definition at line 1105 of file types.f90.

**7.223.2.6 TYPE(EQUATIONS_LINEAR_DATA_TYPE),pointer TYPES::EQUATIONS_-
TYPE::LINEAR_DATA**

A pointer to the data for linear equations.

Definition at line 1107 of file types.f90.

**7.223.2.7 TYPE(EQUATIONS_NONLINEAR_DATA_TYPE),pointer
TYPES::EQUATIONS_TYPE::NONLINEAR_DATA**

A pointer to the data for non-linear equations.

Definition at line 1108 of file types.f90.

7.223.2.8 INTEGER(INTG) TYPES::EQUATIONS_TYPE::OUTPUT_TYPE

The output type for the equations.

See also:

EQUATIONS_ROUTINES_EquationsOutputTypes,EQUATIONS_ROUTINES

Todo

move to equations set???

Definition at line 1103 of file types.f90.

7.223.2.9 INTEGER(INTG) TYPES::EQUATIONS_TYPE::SPARSITY_TYPE

The sparsity type for the equation matrices of the equations.

See also:

EQUATIONS_ROUTINES_EquationsSparsityTypes,EQUATIONS_ROUTINES

Todo

move to equations set???

Definition at line 1104 of file types.f90.

7.223.2.10 TYPE(EQUATIONS_TIME_DATA_TYPE),pointer TYPES::EQUATIONS_- TYPE::TIME_DATA

A pointer to the data for non-static equations.

Definition at line 1109 of file types.f90.

7.224 TYPES::FIELD_CREATE_VALUES_CACHE_TYPE Struct Reference

A type to temporarily hold (cache) the user modifiable values which are used to create a field.

Public Attributes

- INTEGER(INTG) [NUMBER_OF_COMPONENTS](#)

The number of components in the field for each field variable. NOTE: in the future this will need a variable index on it but just allow for the same number of components for each field variable for now.

- INTEGER(INTG), allocatable [VARIABLE_TYPES](#)

VARIABLE_TYPES(variable_idx). The cache of the variable type for the given variable_idx of the field.

- INTEGER(INTG), allocatable [INTERPOLATION_TYPE](#)

INTERPOLATION_TYPES(component_idx,variable_idx). The cache of the interpolation type for the given component and variable of the field.

- INTEGER(INTG), allocatable [MESH_COMPONENT_NUMBER](#)

MESH_COMPONENT_NUMBER(component_idx,variable_idx). The cache of the mesh component number for the given component and variable of the field.

7.224.1 Detailed Description

A type to temporarily hold (cache) the user modifiable values which are used to create a field.

Definition at line 862 of file types.f90.

7.224.2 Member Data Documentation

7.224.2.1 INTEGER(INTG),allocatable TYPES::FIELD_CREATE_VALUES_CACHE_-TYPE::INTERPOLATION_TYPE

INTERPOLATION_TYPES(component_idx,variable_idx). The cache of the interpolation type for the given component and variable of the field.

See also:

[FIELD_ROUTINES::InterpolationTypes](#)

Definition at line 865 of file types.f90.

7.224.2.2 INTEGER(INTG),allocatable TYPES::FIELD_CREATE_VALUES_CACHE_-TYPE::MESH_COMPONENT_NUMBER

MESH_COMPONENT_NUMBER(component_idx,variable_idx). The cache of the mesh component number for the given component and variable of the field.

Definition at line 866 of file types.f90.

7.224.2.3 INTEGER(INTG) TYPES::FIELD_CREATE_VALUES_CACHE_TYPE::NUMBER_OF_COMPONENTS

The number of components in the field for each field variable. NOTE: in the future this will need a variable index on it but just allow for the same number of components for each field variable for now.

Definition at line 863 of file types.f90.

7.224.2.4 INTEGER(INTG),allocatable TYPES::FIELD_CREATE_VALUES_CACHE_TYPE::VARIABLE_TYPES

VARIABLE_TYPES(variable_idx). The cache of the variable type for the given variable_idx of the field.

See also:

[FIELD_ROUTINES::VariableTypes](#)

Definition at line 864 of file types.f90.

7.225 TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE Struct Reference

A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).

Public Attributes

- INTEGER(INTG) [NUMBER_OF_DOFS](#)

The number of degrees-of-freedom for the field.

- INTEGER(INTG), allocatable [DOF_TYPE](#)

DOF_TYPE($i=1..2,ny$). The parameter type of the ny 'th dof. When $i=1$ the DOF_TYPE is the type of the dof, i.e., 1=constant field dof, 2=element based field dof, 3=node based field dof, 4=point based field dof. When $i=2$ the DOF_TYPE gives the nyy 'th number of the different field types and is used to index the XXX_DOF2PARAM_MAP arrays.

- INTEGER(INTG), allocatable [VARIABLE_DOF](#)

VARIABLE_DOF(ny). The variable dof number for the field ny .

- INTEGER(INTG) [NUMBER_OF_CONSTANT_DOFS](#)

The number of constant degrees-of-freedom in the field dofs.

- INTEGER(INTG) [NUMBER_OF_ELEMENT_DOFS](#)

The number of element based degrees-of-freedom in the field dofs.

- INTEGER(INTG) [NUMBER_OF_NODE_DOFS](#)

The number of node based degrees-of-freedom in the field dofs.

- INTEGER(INTG) [NUMBER_OF_POINT_DOFS](#)

The number of point based degrees-of-freedom in the field dofs.

- INTEGER(INTG), allocatable [CONSTANT_DOF2PARAM_MAP](#)

CONSTANT_DOF2PARAM_MAP($i=1..2,nyy$). The mapping from constant field dofs to field parameters for the nyy 'th constant field dof. When $i=1$ the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When $i=2$ the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

- INTEGER(INTG), allocatable [ELEMENT_DOF2PARAM_MAP](#)

ELEMENT_DOF2PARAM_MAP($i=1..3,nyy$). The mapping from element based field dofs to field parameters for the nyy 'th constant field dof. When $i=1$ the DOF2PARAM_MAP gives the element number (ne) of the field parameter. When $i=2$ the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When $i=3$ the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

- INTEGER(INTG), allocatable [NODE_DOF2PARAM_MAP](#)

NODE_DOF2PARAM_MAP($i=1..4,nyy$). The mapping from node based field dofs to field parameters for the nyy 'th constant field dof. When $i=1$ the DOF2PARAM_MAP gives the derivative number (nk) of the field parameter. When $i=2$ the DOF2PARAM_MAP gives the node number (np) of the field parameter. When $i=3$ the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When $i=4$ the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

- INTEGER(INTG), allocatable [POINT_DOF2PARAM_MAP](#)

POINT_DOF2PARAM_MAP(i=1..3,nyy). The mapping from point based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM_MAP gives the point number (nq) of the field parameter. When i=2 the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

7.225.1 Detailed Description

A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).

Definition at line 797 of file types.f90.

7.225.2 Member Data Documentation

7.225.2.1 INTEGER(INTG),allocatable TYPES::FIELD_DOF_TO_PARAM_MAP_- TYPE::CONSTANT_DOF2PARAM_MAP

CONSTANT_DOF2PARAM_MAP(i=1..2,nyy). The mapping from constant field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When i=2 the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

Definition at line 805 of file types.f90.

7.225.2.2 INTEGER(INTG),allocatable TYPES::FIELD_DOF_TO_PARAM_MAP_- TYPE::DOF_TYPE

DOF_TYPE(i=1..2,ny). The parameter type of the ny'th dof. When i=1 the DOF_TYPE is the type of the dof, i.e., 1=constant field dof, 2=element based field dof, 3=node based field dof, 4=point based field dof. When i=2 the DOF_TYPE gives the nyy'th number of the different field types and is used to index the XXX_DOF2PARAM_MAP arrays.

Definition at line 799 of file types.f90.

7.225.2.3 INTEGER(INTG),allocatable TYPES::FIELD_DOF_TO_PARAM_MAP_- TYPE::ELEMENT_DOF2PARAM_MAP

ELEMENT_DOF2PARAM_MAP(i=1..3,nyy). The mapping from element based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM_MAP gives the element number (ne) of the field parameter. When i=2 the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

Definition at line 806 of file types.f90.

**7.225.2.4 INTEGER(INTG),allocatable TYPES::FIELD_DOF_TO_PARAM_MAP_-
TYPE::NODE_DOF2PARAM_MAP**

NODE_DOF2PARAM_MAP(i=1..4,nyy). The mapping from node based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM_MAP gives the derivative number (nk) of the field parameter. When i=2 the DOF2PARAM_MAP gives the node number (np) of the field parameter. When i=3 the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When i=4 the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

Definition at line 807 of file types.f90.

**7.225.2.5 INTEGER(INTG) TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE::NUMBER_OF_-
CONSTANT_DOFS**

The number of constant degrees-of-freedom in the field dofs.

Definition at line 801 of file types.f90.

**7.225.2.6 INTEGER(INTG) TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE::NUMBER_OF_-
DOFS**

The number of degrees-of-freedom for the field.

Definition at line 798 of file types.f90.

**7.225.2.7 INTEGER(INTG) TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE::NUMBER_OF_-
ELEMENT_DOFS**

The number of element based degrees-of-freedom in the field dofs.

Definition at line 802 of file types.f90.

**7.225.2.8 INTEGER(INTG) TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE::NUMBER_OF_-
NODE_DOFS**

The number of node based degrees-of-freedom in the field dofs.

Definition at line 803 of file types.f90.

**7.225.2.9 INTEGER(INTG) TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE::NUMBER_OF_-
POINT_DOFS**

The number of point based degrees-of-freedom in the field dofs.

Definition at line 804 of file types.f90.

**7.225.2.10 INTEGER(INTG),allocatable TYPES::FIELD_DOF_TO_PARAM_MAP_-
TYPE::POINT_DOF2PARAM_MAP**

POINT_DOF2PARAM_MAP(i=1..3,nyy). The mapping from point based field dofs to field parameters for the nyy'th constant field dof. When i=1 the DOF2PARAM_MAP gives the point number (nq) of the field

parameter. When i=2 the DOF2PARAM_MAP gives the component number (nh) of the field parameter. When i=3 the DOF2PARAM_MAP gives the variable number (nc) of the field parameter. The nyy value for a particular field dof (ny) is given by the DOF_TYPE component of this type.

Definition at line 808 of file types.f90.

7.225.2.11 INTEGER(INTG),allocatable TYPES::FIELD_OF_TO_PARAM_MAP_- TYPE::VARIABLE_OF

VARIABLE_OF(ny). The variable dof number for the field ny.

Definition at line 800 of file types.f90.

7.226 TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE Struct Reference

Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.

Collaboration diagram for TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_OF_LINES](#)
The number of lines in the field.
- INTEGER(INTG) [NUMBER_OF AREAS](#)
The number of areas in the field.
- INTEGER(INTG) [NUMBER_OF_VOLUMES](#)
The number of volumes in the field. Inherited from the field decomposition.
- REAL(DP), allocatable [LENGTHS](#)
LENGTHS(nl). The length of the nl'th line in the field decomposition.
- REAL(DP), allocatable [AREAS](#)
AREAS(nf). The area of the nf'th face in the field decomposition.
- REAL(DP), allocatable [VOLUMES](#)
VOLUMES(ne). The volume of the ne'th element in the field decomposition.
- INTEGER(INTG) [NUMBER_OF_FIELDS_USING](#)
The number of fields that use these geometric parameters for their scaling.
- TYPE(FIELD_PTR_TYPE), pointer [FIELDS_USING](#)
FIELDS_USINGS(field_idx). A pointer to the field_idx'th field that uses these geometric parameters for its scaling.

7.226.1 Detailed Description

Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.

Definition at line 770 of file types.f90.

7.226.2 Member Data Documentation

7.226.2.1 REAL(DP),allocatable TYPES::FIELD_GEOMETRIC_PARAMETERS_- TYPE::AREAS

AREAS(nf). The area of the nf'th face in the field decomposition.

Definition at line 775 of file types.f90.

**7.226.2.2 TYPE(FIELD_PTR_TYPE),pointer TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::FIELDS_USING**

FIELDS_USINGS(field_idx). A pointer to the field_idx'th field that uses these geometric parameters for its scaling.

Definition at line 778 of file types.f90.

**7.226.2.3 REAL(DP),allocatable TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::LENGTHS**

LENGTHS(nl). The length of the nl'th line in the field decomposition.

Definition at line 774 of file types.f90.

**7.226.2.4 INTEGER(INTG) TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::NUMBER_OF AREAS**

The number of areas in the field.

Definition at line 772 of file types.f90.

**7.226.2.5 INTEGER(INTG) TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::NUMBER_OF_FIELDS_USING**

The number of fields that use these geometric parameters for their scaling.

Definition at line 777 of file types.f90.

**7.226.2.6 INTEGER(INTG) TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::NUMBER_OF_LINES**

The number of lines in the field.

Definition at line 771 of file types.f90.

**7.226.2.7 INTEGER(INTG) TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::NUMBER_OF_VOLUMES**

The number of volumes in the field. Inherited from the field decomposition.

Definition at line 773 of file types.f90.

**7.226.2.8 REAL(DP),allocatable TYPES::FIELD_GEOMETRIC_PARAMETERS_-
TYPE::VOLUMES**

VOLUMES(ne). The volume of the ne'th element in the field decomposition.

Definition at line 776 of file types.f90.

7.227 TYPES::FIELD_INTERPOLATED_POINT_METRICS_- TYPE Struct Reference

Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.

Collaboration diagram for TYPES::FIELD_INTERPOLATED_POINT_METRICS_TYPE:

Public Attributes

- TYPE(FIELD_INTERPOLATED_POINT_TYPE), pointer INTERPOLATED_POINT
A pointer to the interpolated point.
- INTEGER(INTG) NUMBER_OF_X_DIMENSIONS
The number of X dimensions.
- INTEGER(INTG) NUMBER_OF_XI_DIMENSIONS
The number of Xi dimensions.
- REAL(DP), allocatable GL
GL(mi,ni). Covariant metric tensor. Old CMISS name GL.
- REAL(DP), allocatable GU
GU(mi,ni). Contravariant metric tensor. Old CMISS name GU.
- REAL(DP), allocatable DX_DXI
DX_DXI(nj,ni). Rate of change of the X coordinate system wrt the x coordinate system.
- REAL(DP), allocatable DXI_DX
DXI_DX(ni,nj). Rate of change of the Xi coordinate system wrt the x coordinate system.
- REAL(DP) JACOBIAN
The Jacobian of the Xi to X coordinate system transformation. Old CMISS name RG.
- INTEGER(INTG) JACOBIAN_TYPE
The type of Jacobian.

7.227.1 Detailed Description

Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.

Definition at line 740 of file types.f90.

7.227.2 Member Data Documentation

7.227.2.1 REAL(DP),allocatable TYPES::FIELD_INTERPOLATED_POINT_METRICS_- TYPE::DX_DXI

DX_DXI(nj,ni). Rate of change of the X coordinate system wrt the x coordinate system.

Definition at line 746 of file types.f90.

**7.227.2.2 REAL(DP),allocatable TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::DXI_DX**

DXI_DX(ni,nj). Rate of change of the Xi coordinate system wrt the x coordinate system.

Definition at line 747 of file types.f90.

**7.227.2.3 REAL(DP),allocatable TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::GL**

GL(mi,ni). Covariant metric tensor. Old [CMISS](#) name GL.

Definition at line 744 of file types.f90.

**7.227.2.4 REAL(DP),allocatable TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::GU**

GU(mi,ni). Contravariant metric tensor. Old [CMISS](#) name GU.

Definition at line 745 of file types.f90.

**7.227.2.5 TYPE(FIELD_INTERPOLATED_POINT_TYPE),pointer TYPES::FIELD_-
INTERPOLATED_POINT_METRICS_TYPE::INTERPOLATED_POINT**

A pointer to the interpolated point.

Definition at line 741 of file types.f90.

**7.227.2.6 REAL(DP) TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::JACOBIAN**

The Jacobian of the Xi to X coordinate system transformation. Old [CMISS](#) name RG.

Definition at line 748 of file types.f90.

**7.227.2.7 INTEGER(INTG) TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::JACOBIAN_TYPE**

The type of Jacobian.

See also:

[COORDINATE_ROUTINES::JacobianType](#)

Definition at line 749 of file types.f90.

**7.227.2.8 INTEGER(INTG) TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::NUMBER_OF_X_DIMENSIONS**

The number of X dimensions.

Definition at line 742 of file types.f90.

**7.227.2.9 INTEGER(INTG) TYPES::FIELD_INTERPOLATED_POINT_METRICS_-
TYPE::NUMBER_OF_XI_DIMENSIONS**

The number of Xi dimensions.

Definition at line 743 of file types.f90.

7.228 TYPES::FIELD_INTERPOLATED_POINT_TYPE Struct Reference

Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old CMISS name XG.

Collaboration diagram for TYPES::FIELD_INTERPOLATED_POINT_TYPE:

Public Attributes

- TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE), pointer INTERPOLATION_PARAMETERS
A pointer to the interpolation parameters of the field that is to be interpolated.
- INTEGER(INTG) MAX_PARTIAL_DERIVATIVE_INDEX
The maximum number of partial derivatives that have been allocated for the values component.
- INTEGER(INTG) PARTIAL_DERIVATIVE_TYPE
The type of the partial derivatives that have been interpolated. PARTIAL_DERIVATIVE_TYPE can be either NO_PART_DERIV, FIRST_PART_DERIV or SECOND_PART_DERIV depending on whether just the field value, the field value and all first derivatives (including cross derivatives) or the first value and all first and second derivatives have been interpolated.
- REAL(DP), allocatable VALUES
VALUES(component_idx,nu). The interpolated field components and their partial derivatives.

7.228.1 Detailed Description

Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old CMISS name XG.

Definition at line 753 of file types.f90.

7.228.2 Member Data Documentation

7.228.2.1 TYPE(FIELD_INTERPOLATION_PARAMETERS_TYPE),pointer TYPES::FIELD_INTERPOLATED_POINT_TYPE::INTERPOLATION_PARAMETERS

A pointer to the interpolation parameters of the field that is to be interpolated.

Definition at line 754 of file types.f90.

7.228.2.2 INTEGER(INTG) TYPES::FIELD_INTERPOLATED_POINT_TYPE::MAX_PARTIAL_DERIVATIVE_INDEX

The maximum number of partial derivatives that have been allocated for the values component.

Definition at line 755 of file types.f90.

7.228.2.3 INTEGER(INTG) TYPES::FIELD_INTERPOLATED_POINT_TYPE::PARTIAL_DERIVATIVE_TYPE

The type of the partial derivatives that have been interpolated. PARTIAL_DERIVATIVE_TYPE can be either NO_PART_DERIV, FIRST_PART_DERIV or SECOND_PART_DERIV depending on whether just the field value, the field value and all first derivatives (including cross derivatives) or the first value and all first and second derivatives have been interpolated.

Definition at line 756 of file types.f90.

7.228.2.4 REAL(DP),allocatable TYPES::FIELD_INTERPOLATED_POINT_TYPE::VALUES

VALUES(component_idx,nu). The interpolated field components and their partial derivatives.

Definition at line 757 of file types.f90.

7.229 TYPES::FIELD_INTERPOLATION_PARAMETERS_- TYPE Struct Reference

Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.

Collaboration diagram for TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE:

Public Attributes

- TYPE([FIELD_TYPE](#)), pointer **FIELD**
A pointer to the field to be interpolated.
- TYPE([FIELD_VARIABLE_TYPE](#)), pointer **FIELD_VARIABLE**
A pointer to the field VARIABLE to be interpolated.
- TYPE([BASIS_PTR_TYPE](#)), allocatable **BASES**
BASES(component_idx). An array to hold a pointer to the basis (if any) used for interpolating the component_idx'th component of the field variable.
- INTEGER([INTG](#)), allocatable **NUMBER_OF_PARAMETERS**
NUMBER_OF_PARAMETERS(component_idx). The number of interpolation parameters used for interpolating the component_idx'th component of the field variable.
- REAL([DP](#)), allocatable **PARAMETERS**
PARAMETERS(ns,component_idx). The interpolation parameters used for interpolating the component_idx'th component of the field variable.

7.229.1 Detailed Description

Contains the parameters required to interpolate a field variable within an element. Old [CMISS](#) name XE.

Definition at line 761 of file types.f90.

7.229.2 Member Data Documentation

7.229.2.1 TYPE(BASIS_PTR_TYPE),allocatable TYPES::FIELD_INTERPOLATION_- PARAMETERS_TYPE::BASES

BASES(component_idx). An array to hold a pointer to the basis (if any) used for interpolating the component_idx'th component of the field variable.

Definition at line 764 of file types.f90.

7.229.2.2 TYPE(FIELD_TYPE),pointer TYPES::FIELD_INTERPOLATION_PARAMETERS_- TYPE::FIELD

A pointer to the field to be interpolated.

Definition at line 762 of file types.f90.

**7.229.2.3 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::FIELD_INTERPOLATION_-
PARAMETERS_TYPE::FIELD_VARIABLE**

A pointer to the field VARIABLE to be interpolated.

Definition at line 763 of file types.f90.

**7.229.2.4 INTEGER(INTG),allocatable TYPES::FIELD_INTERPOLATION_PARAMETERS_-
TYPE::NUMBER_OF_PARAMETERS**

NUMBER_OF_PARAMETERS(component_idx). The number of interpolation parameters used for interpolating the component_idx'th component of the field variable.

Definition at line 765 of file types.f90.

**7.229.2.5 REAL(DP),allocatable TYPES::FIELD_INTERPOLATION_PARAMETERS_-
TYPE::PARAMETERS**

PARAMETERS(ns,component_idx). The interpolation parameters used for interpolating the component_idx'th component of the field variable.

Definition at line 766 of file types.f90.

7.230 TYPES::FIELD_MAPPINGS_TYPE Struct Reference

The type containing the mappings for the field.

Collaboration diagram for TYPES::FIELD_MAPPINGS_TYPE:

Public Attributes

- TYPE(FIELD_OF_TO_PARAM_MAP_TYPE) DOF_TO_PARAM_MAP
The mappings for the field dofs to the field parameters.
- TYPE(DOMAIN_MAPPING_TYPE), pointer DOMAIN_MAPPING
The domain mappings for the field dofs i.e., the local to global mpapings etc.

7.230.1 Detailed Description

The type containing the mappings for the field.

Definition at line 870 of file types.f90.

7.230.2 Member Data Documentation

7.230.2.1 TYPE(FIELD_OF_TO_PARAM_MAP_TYPE) TYPES::FIELD_MAPPINGS_- TYPE::DOF_TO_PARAM_MAP

The mappings for the field dofs to the field parameters.

Definition at line 871 of file types.f90.

7.230.2.2 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::FIELD_MAPPINGS_- TYPE::DOMAIN_MAPPING

The domain mappings for the field dofs i.e., the local to global mpapings etc.

Definition at line 872 of file types.f90.

7.231 TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE Struct Reference

A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

Public Attributes

- INTEGER(INTG) [NUMBER_OF_CONSTANT_PARAMETERS](#)

The number of constant field parameters for this field variable component. Note: this is currently always 1 but is.

- INTEGER(INTG) [NUMBER_OF_ELEMENT_PARAMETERS](#)

The number of element based field parameters for this field variable component.

- INTEGER(INTG) [NUMBER_OF_NODE_PARAMETERS](#)

The number of node based field parameters for this field variable component.

- INTEGER(INTG) [MAX_NUMBER_OF_DERIVATIVES](#)

The maximum number of derivatives for the node parameters for this field variable component. It is the size of the first index of the NODE_PARAM2DOF_MAP component of this type.

- INTEGER(INTG) [NUMBER_OF_POINT_PARAMETERS](#)

The number of point based field parameters for this field variable component.

- INTEGER(INTG) [CONSTANT_PARAM2DOF_MAP](#)

CONSTANT_PARAM2DOF_MAP(nc). The field dof (nc=0) or variable dof (nc=1) number of the constant parameter for this field variable component.

- INTEGER(INTG), allocatable [ELEMENT_PARAM2DOF_MAP](#)

ELEMENT_PARAM2DOF_MAP(ne,nc). The field dof (nc=0) or variable dof (nc=1) number of the ne'th element based parameter for this field variable component.

- INTEGER(INTG), allocatable [NODE_PARAM2DOF_MAP](#)

NODE_PARAM2DOF_MAP(nk,np,nc). The field dof (nc=0) or variable dof (nc=1) number of the nk'th derivative of the np'th node based parameter for this field variable component. Note: because the first index of this array is set to the maximum number of derivatives per node this array wastes memory if there are nodes with a smaller number of derivatives than the maximum.

- INTEGER(INTG), allocatable [POINT_PARAM2DOF_MAP](#)

POINT_PARAM2DOF_MAP(nq,nc). The field dof (nc=0) or variable dof (nc=1) number of nq'th point based parameter for this field variable component.

7.231.1 Detailed Description

A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

Definition at line 812 of file types.f90.

7.231.2 Member Data Documentation

7.231.2.1 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::CONSTANT_PARAM2DOF_MAP

CONSTANT_PARAM2DOF_MAP(nc). The field dof (nc=0) or variable dof (nc=1) number of the constant parameter for this field variable component.

Definition at line 819 of file types.f90.

7.231.2.2 INTEGER(INTG),allocatable TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::ELEMENT_PARAM2DOF_MAP

ELEMENT_PARAM2DOF_MAP(ne,nc). The field dof (nc=0) or variable dof (nc=1) number of the ne'th element based parameter for this field variable component.

Todo

Allow for multiple element parameters per element.

Definition at line 820 of file types.f90.

7.231.2.3 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::MAX_NUMBER_OF_DERIVATIVES

The maximum number of derivatives for the node parameters for this field variable component. It is the size of the first index of the NODE_PARAM2DOF_MAP component of this type.

Definition at line 817 of file types.f90.

7.231.2.4 INTEGER(INTG),allocatable TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NODE_PARAM2DOF_MAP

NODE_PARAM2DOF_MAP(nk,np,nc). The field dof (nc=0) or variable dof (nc=1) number of the nk'th derivative of the np'th node based parameter for this field variable component. Note: because the first index of this array is set to the maximum number of derivatives per node this array wastes memory if there are nodes with a smaller number of derivatives than the maximum.

Todo

Don't allocate too much memory if there are different numbers of derivatives for different nodes.

Definition at line 821 of file types.f90.

7.231.2.5 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NUMBER_OF_CONSTANT_PARAMETERS

The number of constant field parameters for this field variable component. Note: this is currently always 1 but is.

Definition at line 813 of file types.f90.

7.231.2.6 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NUMBER_OF_-ELEMENT_PARAMETERS

The number of element based field parameters for this field variable component.

Definition at line 815 of file types.f90.

7.231.2.7 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NUMBER_OF_-NODE_PARAMETERS

The number of node based field parameters for this field variable component.

Definition at line 816 of file types.f90.

7.231.2.8 INTEGER(INTG) TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE::NUMBER_OF_-POINT_PARAMETERS

The number of point based field parameters for this field variable component.

Definition at line 818 of file types.f90.

7.231.2.9 INTEGER(INTG),allocatable TYPES::FIELD_PARAM_TO_DOF_MAP_-TYPE::POINT_PARAM2DOF_MAP

POINT_PARAM2DOF_MAP(nq,nc). The field dof (nc=0) or variable dof (nc=1) number of nq'th point based parameter for this field variable component.

Definition at line 822 of file types.f90.

7.232 TYPES::FIELD_PARAMETER_SET_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a FIELD_PARAMETER_SET_TYPE.

Collaboration diagram for TYPES::FIELD_PARAMETER_SET_PTR_TYPE:

Public Attributes

- TYPE(FIELD_PARAMETER_SET_TYPE), pointer PTR

The pointer to the field parameter set.

7.232.1 Detailed Description

A buffer type to allow for an array of pointers to a FIELD_PARAMETER_SET_TYPE.

Definition at line 884 of file types.f90.

7.232.2 Member Data Documentation

7.232.2.1 TYPE(FIELD_PARAMETER_SET_TYPE),pointer TYPES::FIELD_PARAMETER_SET_PTR_TYPE::PTR

The pointer to the field parameter set.

Definition at line 885 of file types.f90.

7.233 TYPES::FIELD_PARAMETER_SET_TYPE Struct Reference

A type to hold the parameter sets for a field.

Collaboration diagram for TYPES::FIELD_PARAMETER_SET_TYPE:

Public Attributes

- INTEGER(INTG) [SET_INDEX](#)

The global set index (from 1 to the [TYPES::FIELD_PARAMETER_SETS_TYPE::NUMBER_OF_PARAMETER_SETS](#)) that this parameter set corresponds to.

- INTEGER(INTG) [SET_TYPE](#)

The user set type (index) (from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES](#)) that this parameter set.

- TYPE([DISTRIBUTED_VECTOR_TYPE](#)), pointer [PARAMETERS](#)

A pointer to the distributed vector that contains the field parameters for this field parameter set.

7.233.1 Detailed Description

A type to hold the parameter sets for a field.

Definition at line 876 of file types.f90.

7.233.2 Member Data Documentation

7.233.2.1 TYPE([DISTRIBUTED_VECTOR_TYPE](#)),pointer [TYPES::FIELD_PARAMETER_SET_TYPE::PARAMETERS](#)

A pointer to the distributed vector that contains the field parameters for this field parameter set.

Definition at line 880 of file types.f90.

7.233.2.2 INTEGER(INTG) [TYPES::FIELD_PARAMETER_SET_TYPE::SET_INDEX](#)

The global set index (from 1 to the [TYPES::FIELD_PARAMETER_SETS_TYPE::NUMBER_OF_PARAMETER_SETS](#)) that this parameter set corresponds to.

Definition at line 877 of file types.f90.

7.233.2.3 INTEGER(INTG) [TYPES::FIELD_PARAMETER_SET_TYPE::SET_TYPE](#)

The user set type (index) (from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES](#)) that this parameter set.

See also:

[FIELD_ROUTINES::ParameterSetTypes](#)

Definition at line 878 of file types.f90.

7.234 TYPES::FIELD_PARAMETER_SETS_TYPE Struct Reference

A type to store the parameter sets for a field.

Collaboration diagram for TYPES::FIELD_PARAMETER_SETS_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_OF_PARAMETER_SETS](#)

The number of parameter sets that are currently defined on the field.

- TYPE([FIELD_TYPE](#)), pointer [FIELD](#)

A pointer to the field that these parameter sets are defined on.

- TYPE([FIELD_PARAMETER_SET_PTR_TYPE](#)), pointer [SET_TYPE](#)

SET_TYPE(set_type_idx). A pointer to an array of pointers to the field set types. SET_TYPE(set_type_idx)PTR is a pointer to the parameter set type for the set_type_idx'th parameter set. set_type_idx can vary from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES](#). The value of the pointer will be NULL if the parameter set corresponding to the set_type_idx'th parameter set has not yet been created for the field.

- TYPE([FIELD_PARAMETER_SET_PTR_TYPE](#)), pointer [PARAMETER_SETS](#)

PARAMETER_SETS(set_type_idx). A pointer to an array of pointers to the parameter sets that have been created on the field. PARAMETER_SET(set_type_idx)PTR is a pointer to the parameter set type for the set_type_idx'th parameter set that has been created. set_type_idx can vary from 1 to the number of parameter set types that have currently been created for the field i.e., [TYPES::FIELD_PARAMETER_SETS_TYPE::NUMBER_OF_PARAMETER_SETS](#).

7.234.1 Detailed Description

A type to store the parameter sets for a field.

Definition at line 889 of file types.f90.

7.234.2 Member Data Documentation

7.234.2.1 TYPE([FIELD_TYPE](#)),pointer [TYPES::FIELD_PARAMETER_SETS_TYPE::FIELD](#)

A pointer to the field that these parameter sets are defined on.

Definition at line 891 of file types.f90.

7.234.2.2 INTEGER(INTG) [TYPES::FIELD_PARAMETER_SETS_TYPE::NUMBER_OF_PARAMETER_SETS](#)

The number of parameter sets that are currently defined on the field.

Definition at line 890 of file types.f90.

**7.234.2.3 TYPE(FIELD_PARAMETER_SET_PTR_TYPE),pointer
TYPES::FIELD_PARAMETER_SETS_TYPE::PARAMETER_SETS**

PARAMETER_SETS(set_type_idx). A pointer to an array of pointers to the parameter sets that have been created on the field. PARAMETER_SET(set_type_idx)PTR is a pointer to the parameter set type for the set_type_idx'th parameter set that has been created. set_type_idx can vary from 1 to the number of parameter set types that have currently been created for the field i.e., [TYPES::FIELD_PARAMETER_SETS_TYPE::NUMBER_OF_PARAMETER_SETS](#).

Definition at line 893 of file types.f90.

**7.234.2.4 TYPE(FIELD_PARAMETER_SET_PTR_TYPE),pointer
TYPES::FIELD_PARAMETER_SETS_TYPE::SET_TYPE**

SET_TYPE(set_type_idx). A pointer to an array of pointers to the field set types. SET_TYPE(set_type_idx)PTR is a pointer to the parameter set type for the set_type_idx'th parameter set. set_type_idx can vary from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES](#). The value of the pointer will be NULL if the parameter set corresponding to the set_type_idx'th parameter set has not yet been created for the field.

Definition at line 892 of file types.f90.

7.235 TYPES::FIELD_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD_TYPE](#).

Collaboration diagram for TYPES::FIELD_PTR_TYPE:

Public Attributes

- TYPE([FIELD_TYPE](#)), pointer [PTR](#)

The pointer to the field.

7.235.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD_TYPE](#).

Definition at line 919 of file types.f90.

7.235.2 Member Data Documentation

7.235.2.1 TYPE([FIELD_TYPE](#)),pointer TYPES::FIELD_PTR_TYPE::PTR

The pointer to the field.

Definition at line 920 of file types.f90.

7.236 TYPES::FIELD_SCALING_TYPE Struct Reference

A type to hold the scale factors for the appropriate mesh component of a field.

Collaboration diagram for TYPES::FIELD_SCALING_TYPE:

Public Attributes

- INTEGER(INTG) [MESH_COMPONENT_NUMBER](#)

The mesh component number of a field variable component that the scaling factors are associated with.

- INTEGER(INTG) [MAX_NUMBER_OF_DERIVATIVES](#)

The maximum number of derivatives in the mesh component.

- INTEGER(INTG) [MAX_NUMBER_OF_ELEMENT_PARAMETERS](#)

The maximum number of element parameters in the mesh component.

- TYPE([DISTRIBUTED_VECTOR_TYPE](#)), pointer [SCALE_FACTORS](#)

SCALE_FACTORS(nk,np). The scale factor that is applied to the nk'th derivative of the np'th node of the mesh component.

7.236.1 Detailed Description

A type to hold the scale factors for the appropriate mesh component of a field.

Definition at line 782 of file types.f90.

7.236.2 Member Data Documentation

7.236.2.1 INTEGER(INTG) TYPES::FIELD_SCALING_TYPE::MAX_NUMBER_OF_DERIVATIVES

The maximum number of derivatives in the mesh component.

Definition at line 784 of file types.f90.

7.236.2.2 INTEGER(INTG) TYPES::FIELD_SCALING_TYPE::MAX_NUMBER_OF_ELEMENT_PARAMETERS

The maximum number of element parameters in the mesh component.

Definition at line 785 of file types.f90.

7.236.2.3 INTEGER(INTG) TYPES::FIELD_SCALING_TYPE::MESH_COMPONENT_NUMBER

The mesh component number of a field variable component that the scaling factors are associated with.

Definition at line 783 of file types.f90.

**7.236.2.4 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::FIELD_SCALING_-
TYPE::SCALE_FACTORS**

SCALE_FACTORS(nk,np). The scale factor that is applied to the nk'th derivative of the np'th node of the mesh component.

Todo

Make scale factors nodally based for now. Will have to revert to element based and extended to be a matrix to allow for a global derivative to be mapped onto many different element derivatives at the points that closes meshes or for inconsistent xi directions

Definition at line 786 of file types.f90.

7.237 TYPES::FIELD_SCALINGS_TYPE Struct Reference

A type to hold the field scalings for the field.

Collaboration diagram for TYPES::FIELD_SCALINGS_TYPE:

Public Attributes

- INTEGER(INTG) [SCALING_TYPE](#)

The type of scaling that is applied to the field.

- INTEGER(INTG) [NUMBER_OF_SCALING_INDICES](#)

The number of scaling indices (or sets of scale factors) for the field. In general there will be a set of scale factors (or a scaling index) for each different mesh component that is used by the field variable components.

- TYPE([FIELD_SCALING_TYPE](#)), allocatable [SCALINGS](#)

SCALINGS(scaling_idx). The scaling factors for the scaling_idx'th set of scaling factors.

7.237.1 Detailed Description

A type to hold the field scalings for the field.

Definition at line 790 of file types.f90.

7.237.2 Member Data Documentation

7.237.2.1 INTEGER(INTG) TYPES::FIELD_SCALINGS_TYPE::NUMBER_OF_SCALING_INDICES

The number of scaling indices (or sets of scale factors) for the field. In general there will be a set of scale factors (or a scaling index) for each different mesh component that is used by the field variable components.

Definition at line 792 of file types.f90.

7.237.2.2 INTEGER(INTG) TYPES::FIELD_SCALINGS_TYPE::SCALING_TYPE

The type of scaling that is applied to the field.

See also:

[FIELD_ROUTINES::ScalingTypes](#)

Definition at line 791 of file types.f90.

7.237.2.3 TYPE(FIELD_SCALING_TYPE),allocatable TYPES::FIELD_SCALINGS_TYPE::SCALINGS

SCALINGS(scaling_idx). The scaling factors for the scaling_idx'th set of scaling factors.

Definition at line 793 of file types.f90.

7.238 TYPES::FIELD_TYPE Struct Reference

Contains information for a field defined on a region.

Collaboration diagram for TYPES::FIELD_TYPE:

Public Attributes

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The global number of the field in the list of fields for a region.

- INTEGER(INTG) [USER_NUMBER](#)

The user defined identifier for the field. The user number must be unique.

- LOGICAL [FIELD_FINISHED](#)

Is .TRUE. if the field has finished being created, .FALSE. if not.

- TYPE(FIELDS_TYPE), pointer [FIELDS](#)

A pointer to the fields for this region.

- TYPE(REGION_TYPE), pointer [REGION](#)

A pointer to the region for this field.

- INTEGER(INTG) [TYPE](#)

The type of the field.

- INTEGER(INTG) [DEPENDENT_TYPE](#)

The dependent type of the field.

- INTEGER(INTG) [DIMENSION](#)

The dimension of the field.

- TYPE(DECOMPOSITION_TYPE), pointer [DECOMPOSITION](#)

A pointer to the decomposition of the mesh for which the field is defined on.

- INTEGER(INTG) [NUMBER_OF_VARIABLES](#)

The number of variable types in the field. Old CMISS name NCT(nr;nx).

- TYPE(FIELD_VARIABLE_PTR_TYPE), allocatable [VARIABLE_TYPE_MAP](#)

VARIABLE_TYPE_MAP(variable_idx). The map from the available field variable types to the field variable types that are defined for the field. variable_idx varies from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES](#). If the particular field variable type has not been defined on the field then the VARIABLE_TYPE_MAP will be NULL.

- TYPE(FIELD_VARIABLE_TYPE), allocatable [VARIABLES](#)

VARIABLES(variable_idx). The array of field variables.

- TYPE(FIELD_SCALINGS_TYPE) [SCALINGS](#)

The scaling parameters for the field.

- **TYPE(FIELD_MAPPINGS_TYPE) MAPPINGS**

The mappings for the field.

- **TYPE(FIELD_PARAMETER_SETS_TYPE) PARAMETER_SETS**

The parameter sets for the field.

- **TYPE(FIELD_TYPE), pointer GEOMETRIC_FIELD**

A pointer to the geometric field that this field uses. If the field itself is a geometric field then this will be a pointer back to itself.

- **TYPE(FIELD_GEOMETRIC_PARAMETERS_TYPE), pointer GEOMETRIC_FIELD_PARAMETERS**

TYPE(FIELD_CREATE_VALUES_CACHE_TYPE), POINTER :: CREATE_VALUES_CACHE !<The create values cache for the field.

7.238.1 Detailed Description

Contains information for a field defined on a region.

Definition at line 897 of file types.f90.

7.238.2 Member Data Documentation

7.238.2.1 TYPE(DECOMPOSITION_TYPE),pointer TYPES::FIELD_TYPE::DECOMPOSITION

A pointer to the decomposition of the mesh for which the field is defined on.

Definition at line 906 of file types.f90.

7.238.2.2 INTEGER(INTG) TYPES::FIELD_TYPE::DEPENDENT_TYPE

The dependent type of the field.

See also:

[FIELD_ROUTINES::DependentTypes](#)

Definition at line 904 of file types.f90.

7.238.2.3 INTEGER(INTG) TYPES::FIELD_TYPE::DIMENSION

The dimension of the field.

See also:

[FIELD_ROUTINES::DimensionTypes](#)

Definition at line 905 of file types.f90.

7.238.2.4 LOGICAL TYPES::FIELD_TYPE::FIELD_FINISHED

Is .TRUE. if the field has finished being created, .FALSE. if not.

Definition at line 900 of file types.f90.

7.238.2.5 TYPE(FIELDS_TYPE),pointer TYPES::FIELD_TYPE::FIELDS

A pointer to the fields for this region.

Definition at line 901 of file types.f90.

7.238.2.6 TYPE(FIELD_TYPE),pointer TYPES::FIELD_TYPE::GEOMETRIC_FIELD

A pointer to the geometric field that this field uses. If the field itself is a geometric field then this will be a pointer back to itself.

Definition at line 913 of file types.f90.

7.238.2.7 TYPE(FIELD_GEOMETRIC_PARAMETERS_TYPE),pointer TYPES::FIELD_TYPE::GEOMETRIC_FIELD_PARAMETERS

TYPE(FIELD_CREATE_VALUES_CACHE_TYPE), POINTER :: CREATE_VALUES_CACHE !<The create values cache for the field.

Definition at line 914 of file types.f90.

7.238.2.8 INTEGER(INTG) TYPES::FIELD_TYPE::GLOBAL_NUMBER

The global number of the field in the list of fields for a region.

Definition at line 898 of file types.f90.

7.238.2.9 TYPE(FIELD_MAPPINGS_TYPE) TYPES::FIELD_TYPE::MAPPINGS

The mappings for the field.

Definition at line 911 of file types.f90.

7.238.2.10 INTEGER(INTG) TYPES::FIELD_TYPE::NUMBER_OF_VARIABLES

The number of variable types in the field. Old [CMISS](#) name NCT(nr,nx).

Definition at line 907 of file types.f90.

**7.238.2.11 TYPE(FIELD_PARAMETER_SETS_TYPE) TYPES::FIELD_-
TYPE::PARAMETER_SETS**

The parameter sets for the field.

Definition at line 912 of file types.f90.

7.238.2.12 TYPE(REGION_TYPE),pointer TYPES::FIELD_TYPE::REGION

A pointer to the region for this field.

Definition at line 902 of file types.f90.

7.238.2.13 TYPE(FIELD_SCALINGS_TYPE) TYPES::FIELD_TYPE::SCALINGS

The scaling parameters for the field.

Definition at line 910 of file types.f90.

7.238.2.14 INTEGER(INTG) TYPES::FIELD_TYPE::TYPE

The type of the field.

See also:

[FIELD_ROUTINES::FieldTypes](#)

Definition at line 903 of file types.f90.

7.238.2.15 INTEGER(INTG) TYPES::FIELD_TYPE::USER_NUMBER

The user defined identifier for the field. The user number must be unique.

Definition at line 899 of file types.f90.

7.238.2.16 TYPE(FIELD_VARIABLE_PTR_TYPE),allocatable TYPES::FIELD_TYPE::VARIABLE_TYPE_MAP

VARIABLE_TYPE_MAP(variable_idx). The map from the available field variable types to the field variable types that are defined for the field. variable_idx varies from 1 to [FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES](#). If the particular field variable type has not been defined on the field then the VARIABLE_TYPE_MAP will be NULL.

See also:

[FIELD_ROUTINES::VariableTypes](#)

Definition at line 908 of file types.f90.

7.238.2.17 TYPE(FIELD_VARIABLE_TYPE),allocatable TYPES::FIELD_TYPE::VARIABLES

VARIABLES(variable_idx) .The array of field variables.

Definition at line 909 of file types.f90.

7.239 TYPES::FIELD_VARIABLE_COMPONENT_TYPE Struct Reference

Contains information for a component of a field variable.

Collaboration diagram for TYPES::FIELD_VARIABLE_COMPONENT_TYPE:

Public Attributes

- INTEGER(INTG) **COMPONENT_NUMBER**
The number of the field variable component.
- TYPE(FIELD_VARIABLE_TYPE), pointer FIELD_VARIABLE
A pointer to the field variable for this component.
- TYPE(FIELD_TYPE), pointer FIELD
A pointer to the field for this field variable component.
- TYPE(REGION_TYPE), pointer REGION
A pointer to the region for this field variable component.
- INTEGER(INTG) INTERPOLATION_TYPE
The interpolation type of the field variable component.
- INTEGER(INTG) MESH_COMPONENT_NUMBER
The mesh component of the field decomposition for this field variable component.
- INTEGER(INTG) SCALING_INDEX
The index into the defined field scalings for this field variable component.
- TYPE(DOMAIN_TYPE), pointer DOMAIN
A pointer to the domain of the field decomposition for this field variable component.
- INTEGER(INTG) MAX_NUMBER_OF_INTERPOLATION_PARAMETERS
The maximum number of interpolations parameters in an element for a field variable component.
- TYPE(FIELD_PARAM_TO_DOF_MAP_TYPE) PARAM_TO_DOF_MAP
The mapping of the field parameters to the field dofs for this field variable component.

7.239.1 Detailed Description

Contains information for a component of a field variable.

Definition at line 826 of file types.f90.

7.239.2 Member Data Documentation

7.239.2.1 INTEGER(INTG) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::COMPONENT_NUMBER

The number of the field variable component.

Definition at line 827 of file types.f90.

7.239.2.2 TYPE(DOMAIN_TYPE),pointer TYPES::FIELD_VARIABLE_COMPONENT_TYPE::DOMAIN

A pointer to the domain of the field decomposition for this field variable component.

Definition at line 834 of file types.f90.

7.239.2.3 TYPE(FIELD_TYPE),pointer TYPES::FIELD_VARIABLE_COMPONENT_TYPE::FIELD

A pointer to the field for this field variable component.

Definition at line 829 of file types.f90.

7.239.2.4 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::FIELD_VARIABLE_COMPONENT_TYPE::FIELD_VARIABLE

A pointer to the field variable for this component.

Definition at line 828 of file types.f90.

7.239.2.5 INTEGER(INTG) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::INTERPOLATION_TYPE

The interpolation type of the field variable component.

See also:

[FIELD_ROUTINES::InterpolationTypes](#)

Definition at line 831 of file types.f90.

7.239.2.6 INTEGER(INTG) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::MAX_NUMBER_OF_INTERPOLATION_PARAMETERS

The maximum number of interpolations parameters in an element for a field variable component.

Definition at line 835 of file types.f90.

7.239.2.7 INTEGER(INTG) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::MESH_COMPONENT_NUMBER

The mesh component of the field decomposition for this field variable component.

Definition at line 832 of file types.f90.

7.239.2.8 TYPE(FIELD_PARAM_TO_DOF_MAP_TYPE) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::PARAM_TO_DOF_MAP

The mapping of the field parameters to the field dofs for this field variable component.

Definition at line 836 of file types.f90.

7.239.2.9 TYPE(REGION_TYPE),pointer TYPES::FIELD_VARIABLE_COMPONENT_TYPE::REGION

A pointer to the region for this field variable component.

Definition at line 830 of file types.f90.

7.239.2.10 INTEGER(INTG) TYPES::FIELD_VARIABLE_COMPONENT_TYPE::SCALING_INDEX

The index into the defined field scalings for this field variable component.

Definition at line 833 of file types.f90.

7.240 TYPES::FIELD_VARIABLE_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).

Collaboration diagram for TYPES::FIELD_VARIABLE_PTR_TYPE:

Public Attributes

- TYPE([FIELD_VARIABLE_TYPE](#)), pointer PTR

The pointer to the field variable.

7.240.1 Detailed Description

A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).

Definition at line 857 of file types.f90.

7.240.2 Member Data Documentation

7.240.2.1 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::FIELD_VARIABLE_PTR_TYPE::PTR

The pointer to the field variable.

Definition at line 858 of file types.f90.

7.241 TYPES::FIELD_VARIABLE_TYPE Struct Reference

Contains information for a field variable defined on a field.

Collaboration diagram for TYPES::FIELD_VARIABLE_TYPE:

Public Attributes

- INTEGER(INTG) [VARIABLE_NUMBER](#)
The number of the field variable.
- INTEGER(INTG) [VARIABLE_TYPE](#)
The type of the field variable.
- TYPE([FIELD_TYPE](#)), pointer [FIELD](#)
A pointer to the field for this field variable.
- TYPE([REGION_TYPE](#)), pointer [REGION](#)
A pointer to the region for this field variable.
- INTEGER(INTG) [MAX_NUMBER_OF_INTERPOLATION_PARAMETERS](#)
The maximum number of interpolation parameters in an element for a field variable.
- INTEGER(INTG) [NUMBER_OF_DOFS](#)
Number of degrees of freedom for this field variable. Old CMISS name NYNR(0,0,nc,nr,nx).
- INTEGER(INTG) [TOTAL_NUMBER_OF_DOFS](#)
Total number (global) of degrees of freedom for this field variable. Old CMISS name NYNR(0,0,nc,nr,nx).
- INTEGER(INTG) [GLOBAL_DOF_OFFSET](#)
The offset of the start of the global dofs for this variable in the list of global field dofs.
- INTEGER(INTG), allocatable [DOF_LIST](#)
DOF_LIST(i). The list of (local) field dofs in this field variable. Old CMISS name NYNR(1..,0,nc,nr,nx).
- INTEGER(INTG), allocatable [GLOBAL_DOF_LIST](#)
GLOBAL_DOF_LIST(i). The list of global field dofs in this field variable. Old CMISS name NYNR(1..,0,nc,nr,nx).
- TYPE([DOMAIN_MAPPING_TYPE](#)), pointer [DOMAIN_MAPPING](#)
Domain mapping for this variable. Only allocated if the field is a dependent field. May have to reconsider this as we now have a domain mapping for the field as a whole and for each variable of the field. The variable domain mapping to is allow a global matrix/vector mapping to be obtained from the field variable.
- INTEGER(INTG) [NUMBER_OF_COMPONENTS](#)
The number of components in the field variable.
- TYPE([FIELD_VARIABLE_COMPONENT_TYPE](#)), allocatable [COMPONENTS](#)
COMPONENTS(component_idx). The array of field variable components.

7.241.1 Detailed Description

Contains information for a field variable defined on a field.

Definition at line 840 of file types.f90.

7.241.2 Member Data Documentation

7.241.2.1 **TYPE(FIELD_VARIABLE_COMPONENT_TYPE),allocatable TYPES::FIELD_VARIABLE_TYPE::COMPONENTS**

COMPONENTS(component_idx). The array of field variable components.

Definition at line 853 of file types.f90.

7.241.2.2 **INTEGER(INTG),allocatable TYPES::FIELD_VARIABLE_TYPE::DOF_LIST**

DOF_LIST(i). The list of (local) field dofs in this field variable. Old [CMISS](#) name NYNR(1..,0,nc,nr,nx).

Definition at line 849 of file types.f90.

7.241.2.3 **TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::FIELD_VARIABLE_- TYPE::DOMAIN_MAPPING**

Domain mapping for this variable. Only allocated if the field is a dependent field. May have to reconsider this as we now have a domain mapping for the field as a whole and for each variable of the field. The variable domain mapping to is allow a global matrix/vector mapping to be obtained from the field variable.

Definition at line 851 of file types.f90.

7.241.2.4 **TYPE(FIELD_TYPE),pointer TYPES::FIELD_VARIABLE_TYPE::FIELD**

A pointer to the field for this field variable.

Definition at line 843 of file types.f90.

7.241.2.5 **INTEGER(INTG),allocatable TYPES::FIELD_VARIABLE_TYPE::GLOBAL_DOF_- LIST**

GLOBAL_DOF_LIST(i). The list of global field dofs in this field variable. Old [CMISS](#) name NYNR(1..,0,nc,nr,nx).

Definition at line 850 of file types.f90.

7.241.2.6 **INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::GLOBAL_DOF_OFFSET**

The offset of the start of the global dofs for this variable in the list of global field dofs.

Definition at line 848 of file types.f90.

**7.241.2.7 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::MAX_NUMBER_OF_-
INTERPOLATION_PARAMETERS**

The maximum number of interpolation parameters in an element for a field variable.

Definition at line 845 of file types.f90.

**7.241.2.8 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::NUMBER_OF_-
COMPONENTS**

The number of components in the field variable.

Definition at line 852 of file types.f90.

7.241.2.9 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::NUMBER_OF_DOFS

Number of degrees of freedom for this field variable. Old [CMISS](#) name NYNR(0,0,nc,nr,nx).

Definition at line 846 of file types.f90.

7.241.2.10 TYPE(REGION_TYPE),pointer TYPES::FIELD_VARIABLE_TYPE::REGION

A pointer to the region for this field variable.

Definition at line 844 of file types.f90.

**7.241.2.11 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::TOTAL_NUMBER_OF_-
DOFS**

Total number (global) of degrees of freedom for this field variable. Old [CMISS](#) name NYNR(0,0,nc,nr,nx).

Definition at line 847 of file types.f90.

7.241.2.12 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::VARIABLE_NUMBER

The number of the field variable.

Definition at line 841 of file types.f90.

7.241.2.13 INTEGER(INTG) TYPES::FIELD_VARIABLE_TYPE::VARIABLE_TYPE

The type of the field variable.

See also:

[FIELD_ROUTINES::VariableTypes](#)

Definition at line 842 of file types.f90.

7.242 TYPES::FIELDS_TYPE Struct Reference

Contains information on the fields defined on a region.

Collaboration diagram for TYPES::FIELDS_TYPE:

Public Attributes

- TYPE(REGION_TYPE), pointer REGION
A pointer to the region containing the fields.
- INTEGER(INTG) NUMBER_OF_FIELDS
The number of fields defined on the region.
- TYPE(FIELD_PTR_TYPE), pointer FIELDS
FIELDS(fields_idx). The array of pointers to the fields.

7.242.1 Detailed Description

Contains information on the fields defined on a region.

Definition at line 924 of file types.f90.

7.242.2 Member Data Documentation

7.242.2.1 TYPE(FIELD_PTR_TYPE),pointer TYPES::FIELDS_TYPE::FIELDS

FIELDS(fields_idx). The array of pointers to the fields.

Definition at line 927 of file types.f90.

7.242.2.2 INTEGER(INTG) TYPES::FIELDS_TYPE::NUMBER_OF_FIELDS

The number of fields defined on the region.

Definition at line 926 of file types.f90.

7.242.2.3 TYPE(REGION_TYPE),pointer TYPES::FIELDS_TYPE::REGION

A pointer to the region containing the fields.

Definition at line 925 of file types.f90.

7.243 TYPES::GENERATED_MESH_REGULAR_TYPE Struct Reference

Contains information on a generated regular mesh.

Collaboration diagram for TYPES::GENERATED_MESH_REGULAR_TYPE:

Public Attributes

- TYPE(GENERATED_MESH_TYPE), pointer GENERATED_MESH
A pointer to the generated mesh.
- REAL(DP), allocatable ORIGIN
ORIGIN(nj). The position of the origin (first) corner of the regular mesh.
- REAL(DP), allocatable MAXIMUM_EXTENT
MAXIMUM_EXTENT(nj). The extent/size in each nj'th direction of the regular mesh.
- INTEGER(INTG) MESH_DIMENSION
The dimension/number of Xi directions of the regular mesh.
- INTEGER(INTG), allocatable NUMBER_OF_ELEMENTS_XI
NUMBER_OF_ELEMENTS_XI(ni). The number of elements in the ni'th Xi direction for the mesh.
- TYPE(BASIS_TYPE), pointer BASIS
The pointer to the basis used in the regular mesh.

7.243.1 Detailed Description

Contains information on a generated regular mesh.

Definition at line 321 of file types.f90.

7.243.2 Member Data Documentation

7.243.2.1 TYPE(BASIS_TYPE),pointer TYPES::GENERATED_MESH_REGULAR_- TYPE::BASIS

The pointer to the basis used in the regular mesh.

Definition at line 327 of file types.f90.

7.243.2.2 TYPE(GENERATED_MESH_TYPE),pointer TYPES::GENERATED_MESH_- REGULAR_TYPE::GENERATED_MESH

A pointer to the generated mesh.

Definition at line 322 of file types.f90.

**7.243.2.3 REAL(DP),allocatable TYPES::GENERATED_MESH_REGULAR_-
TYPE::MAXIMUM_EXTENT**

MAXIMUM_EXTENT(nj). The extent/size in each nj'th direction of the regular mesh.

Definition at line 324 of file types.f90.

**7.243.2.4 INTEGER(INTG) TYPES::GENERATED_MESH_REGULAR_TYPE::MESH_-
DIMENSION**

The dimension/number of Xi directions of the regular mesh.

Definition at line 325 of file types.f90.

**7.243.2.5 INTEGER(INTG),allocatable TYPES::GENERATED_MESH_REGULAR_-
TYPE::NUMBER_OF_ELEMENTS_XI**

NUMBER_OF_ELEMENTS_XI(ni). The number of elements in the ni'th Xi direction for the mesh.

Definition at line 326 of file types.f90.

7.243.2.6 REAL(DP),allocatable TYPES::GENERATED_MESH_REGULAR_TYPE::ORIGIN

ORIGIN(nj). The position of the origin (first) corner of the regular mesh.

Definition at line 323 of file types.f90.

7.244 TYPES::GENERATED_MESH_TYPE Struct Reference

Contains information on a generated mesh.

Collaboration diagram for TYPES::GENERATED_MESH_TYPE:

Public Attributes

- INTEGER(INTG) [USER_NUMBER](#)
- TYPE([REGION_TYPE](#)), pointer [REGION](#)
- INTEGER(INTG) [GENERATED_TYPE](#)
- TYPE([GENERATED_MESH_REGULAR_TYPE](#)), pointer [REGULAR_MESH](#)
- TYPE([MESH_TYPE](#)), pointer [MESH](#)

7.244.1 Detailed Description

Contains information on a generated mesh.

Definition at line 331 of file types.f90.

7.244.2 Member Data Documentation

7.244.2.1 INTEGER(INTG) TYPES::GENERATED_MESH_TYPE::GENERATED_TYPE

Definition at line 334 of file types.f90.

7.244.2.2 TYPE(MESH_TYPE),pointer TYPES::GENERATED_MESH_TYPE::MESH

Definition at line 336 of file types.f90.

7.244.2.3 TYPE(REGION_TYPE),pointer TYPES::GENERATED_MESH_TYPE::REGION

Definition at line 333 of file types.f90.

7.244.2.4 TYPE(GENERATED_MESH_REGULAR_TYPE),pointer TYPES::GENERATED_MESH_TYPE::REGULAR_MESH

Definition at line 335 of file types.f90.

7.244.2.5 INTEGER(INTG) TYPES::GENERATED_MESH_TYPE::USER_NUMBER

Definition at line 332 of file types.f90.

7.245 TYPES::LINEAR_DIRECT_SOLVER_TYPE Struct Reference

Contains information for a direct linear solver.

Collaboration diagram for TYPES::LINEAR_DIRECT_SOLVER_TYPE:

Public Attributes

- TYPE([LINEAR_SOLVER_TYPE](#)), pointer [LINEAR_SOLVER](#)
A pointer to the linear solver.
- INTEGER(INTG) [SOLVER_LIBRARY](#)
The library type for the linear solver.
- INTEGER(INTG) [DIRECT_SOLVER_TYPE](#)
The type of direct linear solver.

7.245.1 Detailed Description

Contains information for a direct linear solver.

Definition at line 1241 of file types.f90.

7.245.2 Member Data Documentation

7.245.2.1 INTEGER(INTG) TYPES::LINEAR_DIRECT_SOLVER_TYPE::DIRECT_- SOLVER_TYPE

The type of direct linear solver.

Definition at line 1244 of file types.f90.

7.245.2.2 TYPE([LINEAR_SOLVER_TYPE](#)),pointer TYPES::LINEAR_DIRECT_SOLVER_- TYPE::[LINEAR_SOLVER](#)

A pointer to the linear solver.

Definition at line 1242 of file types.f90.

7.245.2.3 INTEGER(INTG) TYPES::LINEAR_DIRECT_SOLVER_TYPE::SOLVER_LIBRARY

The library type for the linear solver.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 1243 of file types.f90.

7.246 TYPES::LINEAR_ITERATIVE_SOLVER_TYPE Struct Reference

Contains information for a direct linear solver.

Collaboration diagram for TYPES::LINEAR_ITERATIVE_SOLVER_TYPE:

Public Attributes

- TYPE([LINEAR_SOLVER_TYPE](#)), pointer [LINEAR_SOLVER](#)
A pointer to the linear solver.
- INTEGER(INTG) [SOLVER_LIBRARY](#)
The library type for the linear solver.
- INTEGER(INTG) [ITERATIVE_SOLVER_TYPE](#)
The type of iterative solver.
- INTEGER(INTG) [ITERATIVE_PRECONDITIONER_TYPE](#)
The type of iterative preconditioner.
- INTEGER(INTG) [MAXIMUM_NUMBER_OF_ITERATIONS](#)
The maximum number of iterations.
- REAL(DP) [RELATIVE_TOLERANCE](#)
The relative tolerance between the rhs and residual norm.
- REAL(DP) [ABSOLUTE_TOLERANCE](#)
The absolute tolerance of the residual norm.
- REAL(DP) [DIVERGENCE_TOLERANCE](#)
The absolute tolerance of the residual norm.
- TYPE([PETSC_PC_TYPE](#)) [PC](#)
The PETSc preconditioner object.
- TYPE([PETSC_KSP_TYPE](#)) [KSP](#)
The PETSc solver object.

7.246.1 Detailed Description

Contains information for a direct linear solver.

Definition at line 1248 of file types.f90.

7.246.2 Member Data Documentation

7.246.2.1 REAL(DP) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::ABSOLUTE_TOLERANCE

The absolute tolerance of the residual norm.

Definition at line 1255 of file types.f90.

7.246.2.2 REAL(DP) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::DIVERGENCE_TOLERANCE

The absolute tolerance of the residual norm.

Definition at line 1256 of file types.f90.

7.246.2.3 INTEGER(INTG) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::ITERATIVE_PRECONDITIONER_TYPE

The type of iterative preconditioner.

Definition at line 1252 of file types.f90.

7.246.2.4 INTEGER(INTG) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::ITERATIVE_SOLVER_TYPE

The type of iterative solver.

Definition at line 1251 of file types.f90.

7.246.2.5 TYPE(PETSC_KSP_TYPE) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::KSP

The PETSc solver object.

Definition at line 1258 of file types.f90.

7.246.2.6 TYPE(LINEAR_SOLVER_TYPE),pointer TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::LINEAR_SOLVER

A pointer to the linear solver.

Definition at line 1249 of file types.f90.

7.246.2.7 INTEGER(INTG) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::MAXIMUM_NUMBER_OF_ITERATIONS

The maximum number of iterations.

Definition at line 1253 of file types.f90.

7.246.2.8 TYPE(PETSC_PC_TYPE) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::PC

The PETSc preconditioner object.

Definition at line 1257 of file types.f90.

7.246.2.9 REAL(DP) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::RELATIVE_TOLERANCE

The relative tolerance between the rhs and residual norm.

Definition at line 1254 of file types.f90.

7.246.2.10 INTEGER(INTG) TYPES::LINEAR_ITERATIVE_SOLVER_TYPE::SOLVER_LIBRARY

The library type for the linear solver.

See also:

[SOLVER_ROUTINES::SolverLibraries](#), [SOLVER_ROUTINES](#)

Definition at line 1250 of file types.f90.

7.247 TYPES::LINEAR_SOLVER_TYPE Struct Reference

Contains information for a linear solver.

Collaboration diagram for TYPES::LINEAR_SOLVER_TYPE:

Public Attributes

- TYPE([SOLVER_TYPE](#)), pointer **SOLVER**
A pointer to the problem_solver.
- INTEGER(INTG) **LINEAR_SOLVER_TYPE**
The type of linear solver.
- TYPE([LINEAR_DIRECT_SOLVER_TYPE](#)), pointer **DIRECT_SOLVER**
A pointer to the direct solver information.
- TYPE([LINEAR_ITERATIVE_SOLVER_TYPE](#)), pointer **ITERATIVE_SOLVER**
A pointer to the iterative solver information.

7.247.1 Detailed Description

Contains information for a linear solver.

Definition at line 1262 of file types.f90.

7.247.2 Member Data Documentation

7.247.2.1 TYPE([LINEAR_DIRECT_SOLVER_TYPE](#)),pointer TYPES::LINEAR_SOLVER_TYPE::DIRECT_SOLVER

A pointer to the direct solver information.

Definition at line 1265 of file types.f90.

7.247.2.2 TYPE([LINEAR_ITERATIVE_SOLVER_TYPE](#)),pointer TYPES::LINEAR_SOLVER_TYPE::ITERATIVE_SOLVER

A pointer to the iterative solver information.

Definition at line 1266 of file types.f90.

7.247.2.3 INTEGER(INTG) TYPES::LINEAR_SOLVER_TYPE::LINEAR_SOLVER_TYPE

The type of linear solver.

See also:

[SOLVER_ROUTINES::LinearSolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 1264 of file types.f90.

7.247.2.4 TYPE(SOLVER_TYPE),pointer TYPES::LINEAR_SOLVER_TYPE::SOLVER

A pointer to the problem_solver.

Definition at line 1263 of file types.f90.

7.248 TYPES::MATRIX_TYPE Struct Reference

Contains information for a matrix.

Public Attributes

- INTEGER(INTG) **ID**
The ID of the matrix.
- LOGICAL **MATRIX_FINISHED**
Is .TRUE. if the matrix has finished being created, .FALSE. if not.
- INTEGER(INTG) **M**
The number of rows in the matrix.
- INTEGER(INTG) **N**
The number of columns in the matrix.
- INTEGER(INTG) **MAX_M**
The maximum number of columns in the matrix storage.
- INTEGER(INTG) **MAX_N**
The maximum number of rows in the matrix storage.
- INTEGER(INTG) **DATA_TYPE**
The data type of the matrix.
- INTEGER(INTG) **STORAGE_TYPE**
The storage type of the matrix.
- INTEGER(INTG) **NUMBER_NON_ZEROS**
The number of non-zero elements in the matrix.
- INTEGER(INTG) **SIZE**
The size of the data arrays.
- INTEGER(INTG) **MAXIMUM_COLUMN_INDICES_PER_ROW**
The maximum number of column indices for the rows.
- INTEGER(INTG), allocatable **ROW_INDICES**
ROW_INDICES(i). The row indices for the matrix storage scheme.
- INTEGER(INTG), allocatable **COLUMN_INDICES**
COLUMN_INDICES(i). The column indices for the matrix storage scheme.
- INTEGER(INTG), allocatable **DATA_INTG**
DATA_INTG(i). The integer data for an integer matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

- REAL(SP), allocatable **DATA_SP**

DATA_SP(i). The real data for a single precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

- REAL(DP), allocatable **DATA_DP**

DATA_DP(i). The real data for a double precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

- LOGICAL, allocatable **DATA_L**

DATA_L(i). The logical data for a logical matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

7.248.1 Detailed Description

Contains information for a matrix.

Definition at line 567 of file types.f90.

7.248.2 Member Data Documentation

7.248.2.1 INTEGER(INTG),allocatable TYPES::MATRIX_TYPE::COLUMN_INDICES

COLUMN_INDICES(i). The column indices for the matrix storage scheme.

See also:

MATRIX_VECTOR_MatrixStorageStructures

Definition at line 580 of file types.f90.

7.248.2.2 REAL(DP),allocatable TYPES::MATRIX_TYPE::DATA_DP

DATA_DP(i). The real data for a double precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 583 of file types.f90.

7.248.2.3 INTEGER(INTG),allocatable TYPES::MATRIX_TYPE::DATA_INTG

DATA_INTG(i). The integer data for an integer matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 581 of file types.f90.

7.248.2.4 LOGICAL,allocatable TYPES::MATRIX_TYPE::DATA_L

DATA_L(i). The logical data for a logical matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 584 of file types.f90.

7.248.2.5 REAL(SP),allocatable TYPES::MATRIX_TYPE::DATA_SP

DATA_SP(i). The real data for a single precision matrix. The i'th component contains the data for the i'th matrix data stored on the domain.

Definition at line 582 of file types.f90.

7.248.2.6 INTEGER(INTG) TYPES::MATRIX_TYPE::DATA_TYPE

The data type of the matrix.

See also:

[MATRIX_VECTOR::DataTypes](#)

Definition at line 574 of file types.f90.

7.248.2.7 INTEGER(INTG) TYPES::MATRIX_TYPE::ID

The ID of the matrix.

Definition at line 568 of file types.f90.

7.248.2.8 INTEGER(INTG) TYPES::MATRIX_TYPE::M

The number of rows in the matrix.

Definition at line 570 of file types.f90.

7.248.2.9 LOGICAL TYPES::MATRIX_TYPE::MATRIX_FINISHED

Is .TRUE. if the matrix has finished being created, .FALSE. if not.

Definition at line 569 of file types.f90.

7.248.2.10 INTEGER(INTG) TYPES::MATRIX_TYPE::MAX_M

The maximum number of columns in the matrix storage.

Definition at line 572 of file types.f90.

7.248.2.11 INTEGER(INTG) TYPES::MATRIX_TYPE::MAX_N

The maximum number of rows in the matrix storage.

Definition at line 573 of file types.f90.

**7.248.2.12 INTEGER(INTG) TYPES::MATRIX_TYPE::MAXIMUM_COLUMN_INDICES_-
PER_ROW**

The maximum number of column indicies for the rows.

Definition at line 578 of file types.f90.

7.248.2.13 INTEGER(INTG) TYPES::MATRIX_TYPE::N

The number of columns in the matrix.

Definition at line 571 of file types.f90.

7.248.2.14 INTEGER(INTG) TYPES::MATRIX_TYPE::NUMBER_NON_ZEROS

The number of non-zero elements in the matrix.

Definition at line 576 of file types.f90.

7.248.2.15 INTEGER(INTG),allocatable TYPES::MATRIX_TYPE::ROW_INDICES

ROW_INDICES(i). The row indices for the matrix storage scheme.

See also:

[MATRIX_VECTOR_MatrixStorageStructures](#)

Definition at line 579 of file types.f90.

7.248.2.16 INTEGER(INTG) TYPES::MATRIX_TYPE::SIZE

The size of the data arrays.

Definition at line 577 of file types.f90.

7.248.2.17 INTEGER(INTG) TYPES::MATRIX_TYPE::STORAGE_TYPE

The storage type of the matrix.

See also:

[MATRIX_VECTOR::StorageTypes](#)

Definition at line 575 of file types.f90.

7.249 TYPES::MESH_DOFS_TYPE Struct Reference

Contains information on the dofs for a mesh.

Collaboration diagram for TYPES::MESH_DOFS_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer MESH
A pointer to the mesh.
- INTEGER(INTG) NUMBER_OF_DOFS
The number of dofs in the mesh.

7.249.1 Detailed Description

Contains information on the dofs for a mesh.

Definition at line 226 of file types.f90.

7.249.2 Member Data Documentation

7.249.2.1 TYPE(MESH_TYPE),pointer TYPES::MESH_DOFS_TYPE::MESH

A pointer to the mesh.

Definition at line 227 of file types.f90.

7.249.2.2 INTEGER(INTG) TYPES::MESH_DOFS_TYPE::NUMBER_OF_DOFS

The number of dofs in the mesh.

Definition at line 228 of file types.f90.

7.250 TYPES::MESH_ELEMENT_TYPE Struct Reference

Contains the information for an element in a mesh.

Collaboration diagram for TYPES::MESH_ELEMENT_TYPE:

Public Attributes

- INTEGER(INTG) [GLOBAL_NUMBER](#)
The global element number in the mesh.
- INTEGER(INTG) [USER_NUMBER](#)
The corresponding user number for the element.
- TYPE([BASIS_TYPE](#)), pointer [BASIS](#)
A pointer to the basis function for the element.
- INTEGER(INTG), allocatable [GLOBAL_ELEMENT_NODES](#)
GLOBAL_ELEMENT_NODES(nn). The global node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).
- INTEGER(INTG), allocatable [USER_ELEMENT_NODES](#)
USER_ELEMENT_NODES(nn). The user node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).
- INTEGER(INTG), allocatable [NUMBER_OF_ADJACENT_ELEMENTS](#)
NUMBER_OF_ADJACENT_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent element before the element in the ni'th direction and +ni gives the adjacent element after the element in the ni'th direction. The ni=0 index should be 1 for the current element. Old CMISS name NXI(-ni:ni,0:nei,ne).
- INTEGER(INTG), allocatable [ADJACENT_ELEMENTS](#)
ADJACENT_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction and +ni gives the adjacent elements after the element in the ni'th direction. The ni=0 index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).

7.250.1 Detailed Description

Contains the information for an element in a mesh.

Definition at line 232 of file types.f90.

7.250.2 Member Data Documentation

7.250.2.1 INTEGER(INTG),allocatable TYPES::MESH_ELEMENT_TYPE::ADJACENT_ELEMENTS

ADJACENT_ELEMENTS(nei,-ni:ni). The local element numbers of the elements adjacent to this element in the ni'th xi direction. Note that -ni gives the adjacent elements before the element in the ni'th direction

and $+ni$ gives the adjacent elements after the element in the ni 'th direction. The $ni=0$ index should give the current element number. Old CMISS name NXI(-ni:ni,0:nei,ne).

Definition at line 239 of file types.f90.

7.250.2.2 TYPE(BASIS_TYPE),pointer TYPES::MESH_ELEMENT_TYPE::BASIS

A pointer to the basis function for the element.

Definition at line 235 of file types.f90.

7.250.2.3 INTEGER(INTG),allocatable TYPES::MESH_ELEMENT_TYPE::GLOBAL_ELEMENT_NODES

GLOBAL_ELEMENT_NODES(nn). The global node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).

Definition at line 236 of file types.f90.

7.250.2.4 INTEGER(INTG) TYPES::MESH_ELEMENT_TYPE::GLOBAL_NUMBER

The global element number in the mesh.

Definition at line 233 of file types.f90.

7.250.2.5 INTEGER(INTG),allocatable TYPES::MESH_ELEMENT_TYPE::NUMBER_OF_ADJACENT_ELEMENTS

NUMBER_OF_ADJACENT_ELEMENTS(-ni:ni). The number of elements adjacent to this element in the ni 'th xi direction. Note that $-ni$ gives the adjacent element before the element in the ni 'th direction and $+ni$ gives the adjacent element after the element in the ni 'th direction. The $ni=0$ index should be 1 for the current element. Old CMISS name NXI(-ni:ni,0:nei,ne).

Definition at line 238 of file types.f90.

7.250.2.6 INTEGER(INTG),allocatable TYPES::MESH_ELEMENT_TYPE::USER_ELEMENT_NODES

USER_ELEMENT_NODES(nn). The user node number in the mesh of the nn'th local node in the element. Old CMISS name NPNE(nn,nbf,ne).

Definition at line 237 of file types.f90.

7.250.2.7 INTEGER(INTG) TYPES::MESH_ELEMENT_TYPE::USER_NUMBER

The corresponding user number for the element.

Definition at line 234 of file types.f90.

7.251 TYPES::MESH_ELEMENTS_TYPE Struct Reference

Contains the information for the elements of a mesh.

Collaboration diagram for TYPES::MESH_ELEMENTS_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer MESH

The pointer to the mesh for the elements information.

- INTEGER(INTG) NUMBER_OF_ELEMENTS

The number of elements in the mesh.

- LOGICAL ELEMENTS_FINISHED

Is .TRUE. if the mesh elements have finished being created, .FALSE. if not.

- TYPE(MESH_ELEMENT_TYPE), pointer ELEMENTS

ELEMENTS(ne). The pointer to the array of information for the elements of this mesh. ELEMENTS(ne) contains the information for the ne'th global element of the mesh.

7.251.1 Detailed Description

Contains the information for the elements of a mesh.

Definition at line 243 of file types.f90.

7.251.2 Member Data Documentation

7.251.2.1 TYPE(MESH_ELEMENT_TYPE),pointer TYPES::MESH_ELEMENTS_TYPE::ELEMENTS

ELEMENTS(ne). The pointer to the array of information for the elements of this mesh. ELEMENTS(ne) contains the information for the ne'th global element of the mesh.

Todo

Should this be allocatable.

Definition at line 247 of file types.f90.

7.251.2.2 LOGICAL TYPES::MESH_ELEMENTS_TYPE::ELEMENTS_FINISHED

Is .TRUE. if the mesh elements have finished being created, .FALSE. if not.

Definition at line 246 of file types.f90.

7.251.2.3 TYPE(MESH_TYPE),pointer TYPES::MESH_ELEMENTS_TYPE::MESH

The pointer to the mesh for the elements information.

Definition at line 244 of file types.f90.

7.251.2.4 INTEGER(INTG) TYPES::MESH_ELEMENTS_TYPE::NUMBER_OF_ELEMENTS

The number of elements in the mesh.

Definition at line 245 of file types.f90.

7.252 TYPES::MESH_NODE_TYPE Struct Reference

Contains the topology information for a global node of a mesh.

Public Attributes

- INTEGER(INTG) [GLOBAL_NUMBER](#)
The global node number in the mesh.
- INTEGER(INTG) [USER_NUMBER](#)
The corresponding user number for the node.
- INTEGER(INTG) [NUMBER_OF_DERIVATIVES](#)
The number of global derivatives at the node for the mesh. Old CMISS name NKT(nj,np).
- INTEGER(INTG), allocatable [PARTIAL_DERIVATIVE_INDEX](#)
PARTIAL_DERIVATIVE_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old CMISS name NUNK(nk,nj,np).
- INTEGER(INTG), allocatable [DOF_INDEX](#)
DOF_INDEX(nk). The global dof derivative index (ny) in the domain of the nk'th global derivative for the node.
- INTEGER(INTG) [NUMBER_OF_SURROUNDING_ELEMENTS](#)
The number of elements surrounding the node in the mesh. Old CMISS name NENP(np,0,0:nr).
- INTEGER(INTG), pointer [SURROUNDING_ELEMENTS](#)
SURROUNDING_ELEMENTS(nep). The global element number of the nep'th element that is surrounding the node. Old CMISS name NENP(np,nep,0:nr).

7.252.1 Detailed Description

Contains the topology information for a global node of a mesh.

Definition at line 251 of file types.f90.

7.252.2 Member Data Documentation

7.252.2.1 INTEGER(INTG),allocatable TYPES::MESH_NODE_TYPE::DOF_INDEX

DOF_INDEX(nk). The global dof derivative index (ny) in the domain of the nk'th global derivative for the node.

Definition at line 256 of file types.f90.

7.252.2.2 INTEGER(INTG) TYPES::MESH_NODE_TYPE::GLOBAL_NUMBER

The global node number in the mesh.

Definition at line 252 of file types.f90.

7.252.2.3 INTEGER(INTG) TYPES::MESH_NODE_TYPE::NUMBER_OF_DERIVATIVES

The number of global derivatives at the node for the mesh. Old [CMISS](#) name NKT(nj,np).

Definition at line 254 of file types.f90.

7.252.2.4 INTEGER(INTG) TYPES::MESH_NODE_TYPE::NUMBER_OF_SURROUNDING_ELEMENTS

The number of elements surrounding the node in the mesh. Old [CMISS](#) name NENP(np,0,0:nr).

Definition at line 257 of file types.f90.

7.252.2.5 INTEGER(INTG),allocatable TYPES::MESH_NODE_TYPE::PARTIAL_DERIVATIVE_INDEX

PARTIAL_DERIVATIVE_INDEX(nk). The partial derivative index (nu) of the nk'th global derivative for the node. Old [CMISS](#) name NUNK(nk,nj,np).

Definition at line 255 of file types.f90.

7.252.2.6 INTEGER(INTG),pointer TYPES::MESH_NODE_TYPE::SURROUNDING_ELEMENTS

SURROUNDING_ELEMENTS(nep). The global element number of the nep'th element that is surrounding the node. Old [CMISS](#) name NENP(np,nep,0:nr).

[Todo](#)

Change this to allocatable.

Definition at line 258 of file types.f90.

7.252.2.7 INTEGER(INTG) TYPES::MESH_NODE_TYPE::USER_NUMBER

The corresponding user number for the node.

Definition at line 253 of file types.f90.

7.253 TYPES::MESH_NODES_TYPE Struct Reference

Contains the information for the nodes of a mesh.

Collaboration diagram for TYPES::MESH_NODES_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer MESH
The pointer to the mesh for this nodes information.
- INTEGER(INTG) NUMBER_OF_NODES
The number of nodes in the mesh.
- TYPE(MESH_NODE_TYPE), pointer NODES
NODES(np). The pointer to the array of topology information for the nodes of the mesh. NODES(np) contains the topological information for the np'th global node of the mesh.

7.253.1 Detailed Description

Contains the information for the nodes of a mesh.

Definition at line 262 of file types.f90.

7.253.2 Member Data Documentation

7.253.2.1 TYPE(MESH_TYPE),pointer TYPES::MESH_NODES_TYPE::MESH

The pointer to the mesh for this nodes information.

Definition at line 263 of file types.f90.

7.253.2.2 TYPE(MESH_NODE_TYPE),pointer TYPES::MESH_NODES_TYPE::NODES

NODES(np). The pointer to the array of topology information for the nodes of the mesh. NODES(np) contains the topological information for the np'th global node of the mesh.

Todo

Should this be allocatable???

Definition at line 265 of file types.f90.

7.253.2.3 INTEGER(INTG) TYPES::MESH_NODES_TYPE::NUMBER_OF_NODES

The number of nodes in the mesh.

Definition at line 264 of file types.f90.

7.254 TYPES::MESH_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a MESH_TYPE.

Collaboration diagram for TYPES::MESH_PTR_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer PTR

The pointer to the mesh.

7.254.1 Detailed Description

A buffer type to allow for an array of pointers to a MESH_TYPE.

Definition at line 303 of file types.f90.

7.254.2 Member Data Documentation

7.254.2.1 TYPE(MESH_TYPE),pointer TYPES::MESH_PTR_TYPE::PTR

The pointer to the mesh.

Definition at line 304 of file types.f90.

7.255 TYPES::MESH_TOPOLOGY_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a MESH_TOPOLOGY_TYPE.

Collaboration diagram for TYPES::MESH_TOPOLOGY_PTR_TYPE:

Public Attributes

- TYPE(MESH_TOPOLOGY_TYPE), pointer PTR
The pointer to the mesh topology.

7.255.1 Detailed Description

A buffer type to allow for an array of pointers to a MESH_TOPOLOGY_TYPE.

Definition at line 278 of file types.f90.

7.255.2 Member Data Documentation

7.255.2.1 TYPE(MESH_TOPOLOGY_TYPE),pointer TYPES::MESH_TOPOLOGY_PTR_- TYPE::PTR

The pointer to the mesh topology.

Definition at line 279 of file types.f90.

7.256 TYPES::MESH_TOPOLOGY_TYPE Struct Reference

Contains information on the (global) topology of a mesh.

Collaboration diagram for TYPES::MESH_TOPOLOGY_TYPE:

Public Attributes

- TYPE(MESH_TYPE), pointer MESH
Pointer to the parent mesh.
- INTEGER(INTG) MESH_COMPONENT_NUMBER
The mesh component number for this mesh topology.
- TYPE(MESH_NODES_TYPE), pointer NODES
Pointer to the nodes within the mesh topology.
- TYPE(MESH_ELEMENTS_TYPE), pointer ELEMENTS
Pointer to the elements within the mesh topology.
- TYPE(MESH_DOFS_TYPE), pointer DOFS
Pointer to the dofs within the mesh topology.

7.256.1 Detailed Description

Contains information on the (global) topology of a mesh.

Definition at line 269 of file types.f90.

7.256.2 Member Data Documentation

7.256.2.1 TYPE(MESH_DOFS_TYPE),pointer TYPES::MESH_TOPOLOGY_TYPE::DOFS

Pointer to the dofs within the mesh topology.

Definition at line 274 of file types.f90.

7.256.2.2 TYPE(MESH_ELEMENTS_TYPE),pointer TYPES::MESH_TOPOLOGY_TYPE::ELEMENTS

Pointer to the elements within the mesh topology.

Definition at line 273 of file types.f90.

7.256.2.3 TYPE(MESH_TYPE),pointer TYPES::MESH_TOPOLOGY_TYPE::MESH

Pointer to the parent mesh.

Definition at line 270 of file types.f90.

7.256.2.4 INTEGER(INTG) TYPES::MESH_TOPOLOGY_TYPE::MESH_COMPONENT_NUMBER

The mesh component number for this mesh topology.

Definition at line 271 of file types.f90.

7.256.2.5 TYPE(MESH_NODES_TYPE),pointer TYPES::MESH_TOPOLOGY_TYPE::NODES

Pointer to the nodes within the mesh topology.

Definition at line 272 of file types.f90.

7.257 TYPES::MESH_TYPE Struct Reference

Contains information on a mesh defined on a region.

Collaboration diagram for TYPES::MESH_TYPE:

Public Attributes

- INTEGER(INTG) [USER_NUMBER](#)
The user number of the mesh. The user number must be unique.
- INTEGER(INTG) [GLOBAL_NUMBER](#)
The corresponding global number for the mesh.
- LOGICAL [MESH_FINISHED](#)
Is .TRUE. if the mesh has finished being created, .FALSE. if not.
- TYPE([MESHS_TYPE](#)), pointer [MESHS](#)
A pointer to the meshes for this mesh.
- TYPE([REGION_TYPE](#)), pointer [REGION](#)
A pointer to the region containing this mesh.
- INTEGER(INTG) [NUMBER_OF_DIMENSIONS](#)
The number of dimensions (X_i directions) for this mesh.
- INTEGER(INTG) [NUMBER_OF_COMPONENTS](#)
The number of mesh components in this mesh.
- LOGICAL [MESH_EMBEDDED](#)
Is .TRUE. if the mesh is embedded in another mesh, .FALSE. if not.
- TYPE([MESH_TYPE](#)), pointer [EMBEDDING_MESH](#)
If this mesh is embedded the pointer to the mesh that this mesh is embedded in. IF the mesh is not embedded the pointer is NULL.
- INTEGER(INTG) [NUMBER_OF_EMBEDDED_MESHES](#)
The number of meshes that are embedded in this mesh.
- TYPE([MESH_PTR_TYPE](#)), pointer [EMBEDDED_MESHES](#)
EMBEDDED_MESHES(mesh_idx). A pointer to the mesh_idx'th mesh that is embedded in this mesh.
- INTEGER(INTG) [NUMBER_OF_ELEMENTS](#)
The number of elements in the mesh.
- INTEGER(INTG) [NUMBER_OF_FACES](#)
The number of faces in the mesh.
- INTEGER(INTG) [NUMBER_OF_LINES](#)
The number of lines in the mesh.

- TYPE([MESH_TOPOLOGY_PTR_TYPE](#)), pointer [TOPOLOGY](#)

TOPOLOGY(mesh_component_idx). A pointer to the topology mesh_component_idx'th mesh component.

- TYPE([DECOMPOSITIONS_TYPE](#)), pointer [DECOMPOSITIONS](#)

A pointer to the decompositions for this mesh.

7.257.1 Detailed Description

Contains information on a mesh defined on a region.

Definition at line 283 of file types.f90.

7.257.2 Member Data Documentation

7.257.2.1 TYPE(DECOMPOSITIONS_TYPE),pointer TYPES::MESH_- TYPE::DECOMPOSITIONS

A pointer to the decompositions for this mesh.

Definition at line 299 of file types.f90.

7.257.2.2 TYPE(MESH_PTR_TYPE),pointer TYPES::MESH_TYPE::EMBEDDED_MESHES

EMBEDDED_MESHES(mesh_idx). A pointer to the mesh_idx'th mesh that is embedded in this mesh.

Definition at line 294 of file types.f90.

7.257.2.3 TYPE(MESH_TYPE),pointer TYPES::MESH_TYPE::EMBEDDING_MESH

If this mesh is embedded the pointer to the mesh that this mesh is embedded in. IF the mesh is not embedded the pointer is NULL.

Definition at line 292 of file types.f90.

7.257.2.4 INTEGER(INTG) TYPES::MESH_TYPE::GLOBAL_NUMBER

The corresponding global number for the mesh.

Definition at line 285 of file types.f90.

7.257.2.5 LOGICAL TYPES::MESH_TYPE::MESH_EMBEDDED

Is .TRUE. if the mesh is embedded in another mesh, .FALSE. if not.

Definition at line 291 of file types.f90.

7.257.2.6 LOGICAL TYPES::MESH_TYPE::MESH_FINISHED

Is .TRUE. if the mesh has finished being created, .FALSE. if not.

Definition at line 286 of file types.f90.

7.257.2.7 TYPE(MESHES_TYPE),pointer TYPES::MESH_TYPE::MESHES

A pointer to the meshes for this mesh.

Definition at line 287 of file types.f90.

7.257.2.8 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_COMPONENTS

The number of mesh components in this mesh.

Definition at line 290 of file types.f90.

7.257.2.9 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_DIMENSIONS

The number of dimensions (Xi directions) for this mesh.

Definition at line 289 of file types.f90.

7.257.2.10 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_ELEMENTS

The number of elements in the mesh.

Definition at line 295 of file types.f90.

7.257.2.11 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_EMBEDDED_MESHES

The number of meshes that are embedded in this mesh.

Definition at line 293 of file types.f90.

7.257.2.12 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_FACES

The number of faces in the mesh.

Definition at line 296 of file types.f90.

7.257.2.13 INTEGER(INTG) TYPES::MESH_TYPE::NUMBER_OF_LINES

The number of lines in the mesh.

Definition at line 297 of file types.f90.

7.257.2.14 TYPE(REGION_TYPE),pointer TYPES::MESH_TYPE::REGION

A pointer to the region containing this mesh.

Definition at line 288 of file types.f90.

**7.257.2.15 TYPE(MESH_TOPOLOGY_PTR_TYPE),pointer TYPES::MESH_-
TYPE::TOPOLOGY**

TOPOLOGY(mesh_component_idx). A pointer to the topology mesh_component_idx'th mesh component.

Definition at line 298 of file types.f90.

7.257.2.16 INTEGER(INTG) TYPES::MESH_TYPE::USER_NUMBER

The user number of the mesh. The user number must be unique.

Definition at line 284 of file types.f90.

7.258 TYPES::MESHES_TYPE Struct Reference

Contains information on the meshes defined on a region.

Collaboration diagram for TYPES::MESHES_TYPE:

Public Attributes

- TYPE([REGION_TYPE](#)), pointer [REGION](#)
A pointer to the region.
- INTEGER(INTG) [NUMBER_OF_MESHES](#)
The number of meshes defined on the region.
- TYPE([MESH_PTR_TYPE](#)), pointer [MESHES](#)
MESHES(meshes_idx). The array of pointers to the meshes.

7.258.1 Detailed Description

Contains information on the meshes defined on a region.

Definition at line 308 of file types.f90.

7.258.2 Member Data Documentation

7.258.2.1 TYPE(MESH_PTR_TYPE),pointer TYPES::MESHES_TYPE::MESHES

MESHES(meshes_idx). The array of pointers to the meshes.

Definition at line 311 of file types.f90.

7.258.2.2 INTEGER(INTG) TYPES::MESHES_TYPE::NUMBER_OF_MESHES

The number of meshes defined on the region.

Definition at line 310 of file types.f90.

7.258.2.3 TYPE(REGION_TYPE),pointer TYPES::MESHES_TYPE::REGION

A pointer to the region.

Definition at line 309 of file types.f90.

7.259 TYPES::NODE_TYPE Struct Reference

Contains information about a node.

Collaboration diagram for TYPES::NODE_TYPE:

Public Attributes

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The global number of node.

- INTEGER(INTG) [USER_NUMBER](#)

The user defined number of node.

- TYPE([VARYING_STRING](#)) [LABEL](#)

A string label for the node.

- REAL(DP), allocatable [INITIAL_POSITION](#)

INITIAL_POSITION(nj). The initial position of the node. Used to identify the position of the node before meshing (e.g., Delauny triangulation) as the geometric field has not been created when the node is created. The actual geometric coordinates for computation using the node should be taken from the geometric field which uses the node.

7.259.1 Detailed Description

Contains information about a node.

Definition at line 203 of file types.f90.

7.259.2 Member Data Documentation

7.259.2.1 INTEGER(INTG) TYPES::NODE_TYPE::GLOBAL_NUMBER

The global number of node.

Definition at line 204 of file types.f90.

7.259.2.2 REAL(DP),allocatable TYPES::NODE_TYPE::INITIAL_POSITION

INITIAL_POSITION(nj). The initial position of the node. Used to identify the position of the node before meshing (e.g., Delauny triangulation) as the geometric field has not been created when the node is created. The actual geometric coordinates for computation using the node should be taken from the geometric field which uses the node.

Definition at line 207 of file types.f90.

7.259.2.3 TYPE(VARYING_STRING) TYPES::NODE_TYPE::LABEL

A string label for the node.

Definition at line 206 of file types.f90.

7.259.2.4 INTEGER(INTG) TYPES::NODE_TYPE::USER_NUMBER

The user defined number of node.

Definition at line 205 of file types.f90.

7.260 TYPES::NODES_TYPE Struct Reference

Contains information on the nodes defined on a region.

Collaboration diagram for TYPES::NODES_TYPE:

Public Attributes

- TYPE(REGION_TYPE), pointer REGION
A pointer to the region containing the nodes.
- INTEGER(INTG) NUMBER_OF_NODES
The number of nodes defined on the region.
- LOGICAL NODES_FINISHED
Is .TRUE. if the nodes have finished being created, .FALSE. if not.
- TYPE(NODE_TYPE), pointer NODES
NODES(nodes_idx). A pointer to the nodes.
- TYPE(TREE_TYPE), pointer NODE_TREE
The tree for user to global node mapping.

7.260.1 Detailed Description

Contains information on the nodes defined on a region.

Definition at line 211 of file types.f90.

7.260.2 Member Data Documentation

7.260.2.1 TYPE(TREE_TYPE),pointer TYPES::NODES_TYPE::NODE_TREE

The tree for user to global node mapping.

Definition at line 216 of file types.f90.

7.260.2.2 TYPE(NODE_TYPE),pointer TYPES::NODES_TYPE::NODES

NODES(nodes_idx). A pointer to the nodes.

Todo

Should this be allocatable?

Definition at line 215 of file types.f90.

7.260.2.3 LOGICAL TYPES::NODES_TYPE::NODES_FINISHED

Is .TRUE. if the nodes have finished being created, .FALSE. if not.

Definition at line 214 of file types.f90.

7.260.2.4 INTEGER(INTG) TYPES::NODES_TYPE::NUMBER_OF_NODES

The number of nodes defined on the region.

Definition at line 213 of file types.f90.

7.260.2.5 TYPE(REGION_TYPE),pointer TYPES::NODES_TYPE::REGION

A pointer to the region containing the nodes.

Definition at line 212 of file types.f90.

7.261 TYPES::NONLINEAR_SOLVER_TYPE Struct Reference

Contains information for a nonlinear solver.

Collaboration diagram for TYPES::NONLINEAR_SOLVER_TYPE:

Public Attributes

- TYPE([SOLVER_TYPE](#)), pointer **SOLVER**

A pointer to the problem_solver.

- INTEGER(INTG) **SOLVER_LIBRARY**

The library type for the nonlinear solver.

7.261.1 Detailed Description

Contains information for a nonlinear solver.

Definition at line 1270 of file types.f90.

7.261.2 Member Data Documentation

7.261.2.1 TYPE(SOLVER_TYPE),pointer TYPES::NONLINEAR_SOLVER_TYPE::SOLVER

A pointer to the problem_solver.

Definition at line 1271 of file types.f90.

7.261.2.2 INTEGER(INTG) TYPES::NONLINEAR_SOLVER_TYPE::SOLVER_LIBRARY

The library type for the nonlinear solver.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 1272 of file types.f90.

7.262 TYPES::PROBLEM_CONTROL_TYPE Struct Reference

Collaboration diagram for TYPES::PROBLEM_CONTROL_TYPE:

Public Attributes

- TYPE([PROBLEM_TYPE](#)), pointer [PROBLEM](#)
A pointer back to the problem.
- LOGICAL [CONTROL_FINISHED](#)
Is .TRUE. if the problem control has finished being created, .FALSE. if not.
- INTEGER(INTG) [CONTROL_TYPE](#)

7.262.1 Detailed Description

Definition at line 1483 of file types.f90.

7.262.2 Member Data Documentation

7.262.2.1 LOGICAL TYPES::PROBLEM_CONTROL_TYPE::CONTROL_FINISHED

Is .TRUE. if the problem control has finished being created, .FALSE. if not.

Definition at line 1485 of file types.f90.

7.262.2.2 INTEGER(INTG) TYPES::PROBLEM_CONTROL_TYPE::CONTROL_TYPE

Definition at line 1486 of file types.f90.

7.262.2.3 TYPE(PROBLEM_TYPE),pointer TYPES::PROBLEM_CONTROL_TYPE::PROBLEM

A pointer back to the problem.

Definition at line 1484 of file types.f90.

7.263 TYPES::PROBLEM_LINEAR_DATA_TYPE Struct Reference

Contains information on any data required for a linear solution.

Collaboration diagram for TYPES::PROBLEM_LINEAR_DATA_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer SOLUTION

A pointer to the problem solution.

7.263.1 Detailed Description

Contains information on any data required for a linear solution.

Definition at line 1468 of file types.f90.

7.263.2 Member Data Documentation

7.263.2.1 TYPE(SOLUTION_TYPE),pointer TYPES::PROBLEM_LINEAR_DATA_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1469 of file types.f90.

7.264 TYPES::PROBLEM_NONLINEAR_DATA_TYPE Struct Reference

Contains information on any data required for a non-linear solution.

Collaboration diagram for TYPES::PROBLEM_NONLINEAR_DATA_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer SOLUTION
A pointer to the problem solution.
- INTEGER(INTG) NUMBER_OF_ITERATIONS

7.264.1 Detailed Description

Contains information on any data required for a non-linear solution.

Definition at line 1473 of file types.f90.

7.264.2 Member Data Documentation

7.264.2.1 INTEGER(INTG) TYPES::PROBLEM_NONLINEAR_DATA_TYPE::NUMBER_OF_ITERATIONS

Definition at line 1475 of file types.f90.

7.264.2.2 TYPE(SOLUTION_TYPE),pointer TYPES::PROBLEM_NONLINEAR_DATA_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1474 of file types.f90.

7.265 TYPES::PROBLEM_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a PROBLEM_TYPE.

Collaboration diagram for TYPES::PROBLEM_PTR_TYPE:

Public Attributes

- TYPE(PROBLEM_TYPE), pointer PTR

The pointer to the problem.

7.265.1 Detailed Description

A buffer type to allow for an array of pointers to a PROBLEM_TYPE.

Definition at line 1507 of file types.f90.

7.265.2 Member Data Documentation

7.265.2.1 TYPE(PROBLEM_TYPE),pointer TYPES::PROBLEM_PTR_TYPE::PTR

The pointer to the problem.

Definition at line 1508 of file types.f90.

7.266 TYPES::PROBLEM_TIME_DATA_TYPE Struct Reference

Contains information on any data required for a time-dependent solution.

Collaboration diagram for TYPES::PROBLEM_TIME_DATA_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer SOLUTION
A pointer to the problem solution.

7.266.1 Detailed Description

Contains information on any data required for a time-dependent solution.

Definition at line 1479 of file types.f90.

7.266.2 Member Data Documentation

7.266.2.1 TYPE(SOLUTION_TYPE),pointer TYPES::PROBLEM_TIME_DATA_- TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1480 of file types.f90.

7.267 TYPES::PROBLEM_TYPE Struct Reference

Contains information for a problem.

Collaboration diagram for TYPES::PROBLEM_TYPE:

Public Attributes

- INTEGER(INTG) **USER_NUMBER**
The user defined identifier for the problem. The user number must be unique.
- INTEGER(INTG) **GLOBAL_NUMBER**
The global number of the problem in the list of problems.
- LOGICAL **PROBLEM_FINISHED**
Is .TRUE. if the problem has finished being created, .FALSE. if not.
- TYPE(PROBLEMS_TYPE), pointer **PROBLEMS**
A pointer to the problems for this problem.
- INTEGER(INTG) **CLASS**
The problem specification class identifier.
- INTEGER(INTG) **TYPE**
The problem specification type identifier.
- INTEGER(INTG) **SUBTYPE**
The problem specification subtype identifier.
- TYPE(PROBLEM_CONTROL_TYPE), pointer **CONTROL**
A pointer to the control information for the problem.
- INTEGER(INTG) **NUMBER_OF_SOLUTIONS**
The number of solutions in the problem.
- TYPE(SOLUTION_PTR_TYPE), allocatable **SOLUTIONS**
A pointer to the solution information for the problem.

7.267.1 Detailed Description

Contains information for a problem.

Definition at line 1490 of file types.f90.

7.267.2 Member Data Documentation

7.267.2.1 INTEGER(INTG) TYPES::PROBLEM_TYPE::CLASS

The problem specification class identifier.

Definition at line 1496 of file types.f90.

7.267.2.2 TYPE (PROBLEM_CONTROL_TYPE),pointer TYPES::PROBLEM_TYPE::CONTROL

A pointer to the control information for the problem.

Definition at line 1500 of file types.f90.

7.267.2.3 INTEGER(INTG) TYPES::PROBLEM_TYPE::GLOBAL_NUMBER

The global number of the problem in the list of problems.

Definition at line 1492 of file types.f90.

7.267.2.4 INTEGER(INTG) TYPES::PROBLEM_TYPE::NUMBER_OF_SOLUTIONS

The number of solutions in the problem.

Definition at line 1502 of file types.f90.

7.267.2.5 LOGICAL TYPES::PROBLEM_TYPE::PROBLEM_FINISHED

Is .TRUE. if the problem has finished being created, .FALSE. if not.

Definition at line 1493 of file types.f90.

7.267.2.6 TYPE(PROBLEMS_TYPE),pointer TYPES::PROBLEM_TYPE::PROBLEMS

A pointer to the problems for this problem.

Definition at line 1494 of file types.f90.

7.267.2.7 TYPE(SOLUTION_PTR_TYPE),allocatable TYPES::PROBLEM_TYPE::SOLUTIONS

A pointer to the solution information for the problem.

Definition at line 1503 of file types.f90.

7.267.2.8 INTEGER(INTG) TYPES::PROBLEM_TYPE::SUBTYPE

The problem specification subtype identifier.

Definition at line 1498 of file types.f90.

7.267.2.9 INTEGER(INTG) TYPES::PROBLEM_TYPE::TYPE

The problem specification type identifier.

Definition at line 1497 of file types.f90.

7.267.2.10 INTEGER(INTG) TYPES::PROBLEM_TYPE::USER_NUMBER

The user defined identifier for the problem. The user number must be unique.

Definition at line 1491 of file types.f90.

7.268 TYPES::PROBLEMS_TYPE Struct Reference

Contains information on the problems defined on a region.

Collaboration diagram for TYPES::PROBLEMS_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_OF_PROBLEMS](#)
The number of problems defined.
- TYPE([PROBLEM_PTR_TYPE](#)), pointer [PROBLEMS](#)
The array of pointers to the problems.

7.268.1 Detailed Description

Contains information on the problems defined on a region.

Definition at line 1512 of file types.f90.

7.268.2 Member Data Documentation

7.268.2.1 INTEGER(INTG) TYPES::PROBLEMS_TYPE::NUMBER_OF_PROBLEMS

The number of problems defined.

Definition at line 1513 of file types.f90.

7.268.2.2 TYPE(PROBLEM_PTR_TYPE),pointer TYPES::PROBLEMS_TYPE::PROBLEMS

The array of pointers to the problems.

Definition at line 1514 of file types.f90.

7.269 TYPES::QUADRATURE_SCHEME_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a QUADRATURE_SCHEME_TYPE.

Collaboration diagram for TYPES::QUADRATURE_SCHEME_PTR_TYPE:

Public Attributes

- TYPE(QUADRATURE_SCHEME_TYPE), pointer PTR
A pointer to the quadrature scheme.

7.269.1 Detailed Description

A buffer type to allow for an array of pointers to a QUADRATURE_SCHEME_TYPE.

See also:

[TYPES::QUADRATURE_SCHEME_TYPE](#)

Definition at line 96 of file types.f90.

7.269.2 Member Data Documentation

7.269.2.1 TYPE(QUADRATURE_SCHEME_TYPE),pointer TYPES::QUADRATURE_SCHEME_PTR_TYPE::PTR

A pointer to the quadrature scheme.

Definition at line 97 of file types.f90.

7.270 TYPES::QUADRATURE_SCHEME_TYPE Struct Reference

Contains information for a particular quadrature scheme.

Collaboration diagram for TYPES::QUADRATURE_SCHEME_TYPE:

Public Attributes

- INTEGER(INTG) [GLOBAL_NUMBER](#)

The global number of the quadrature scheme in the list of quadrature schemes for a particular quadrature.

- TYPE([QUADRATURE_TYPE](#)), pointer [QUADRATURE](#)

The pointer back to the quadrature for a particular quadrature scheme.

- INTEGER(INTG) [NUMBER_OF_GAUSS](#)

The number of gauss points for the quadrature scheme.

- REAL(DP), allocatable [GAUSS_POSITIONS](#)

GAUSS_POSITIONS(nic,ng). The positions in the nic'th xi coordinate of Gauss point ng. Old [CMISS](#) name XIG(ni,ng,nb).

- REAL(DP), allocatable [GAUSS_WEIGHTS](#)

GAUSS_WEIGHTS(ng). The weight applied to Gauss point ng. Old [CMISS](#) name WG(ng,nb).

- REAL(DP), allocatable [GAUSS_BASIS_FNS](#)

GAUSS_BASIS_FNS(ns,nu,ng). The value of the basis functions evaluated at Gauss point ng for the nu'th derivative of the basis function associated with the ns'th element parameter. Old [CMISS](#) name PG(ns,nu,ng,nb).

7.270.1 Detailed Description

Contains information for a particular quadrature scheme.

[Todo](#)

Also evaluate the product of the basis functions at gauss points for speed???

Definition at line 86 of file types.f90.

7.270.2 Member Data Documentation

7.270.2.1 REAL(DP),allocatable TYPES::QUADRATURE_SCHEME_TYPE::GAUSS_BASIS_FNS

GAUSS_BASIS_FNS(ns,nu,ng). The value of the basis functions evaluated at Gauss point ng for the nu'th derivative of the basis function associated with the ns'th element parameter. Old [CMISS](#) name PG(ns,nu,ng,nb).

Definition at line 92 of file types.f90.

7.270.2.2 REAL(DP),allocatable TYPES::QUADRATURE_SCHEME_TYPE::GAUSS_POSITIONS

GAUSS_POSITIONS(nic,ng). The positions in the nic'th xi coordinate of Gauss point ng. Old [CMISS](#) name XIG(ni,ng,nb).

Definition at line 90 of file types.f90.

7.270.2.3 REAL(DP),allocatable TYPES::QUADRATURE_SCHEME_TYPE::GAUSS_WEIGHTS

GAUSS_WEIGHTS(ng). The weight applied to Gauss point ng. Old [CMISS](#) name WG(ng,nb).

Definition at line 91 of file types.f90.

7.270.2.4 INTEGER(INTG) TYPES::QUADRATURE_SCHEME_TYPE::GLOBAL_NUMBER

The global number of the quadrature scheme in the list of quadrature schemes for a particular quadrature.

Definition at line 87 of file types.f90.

7.270.2.5 INTEGER(INTG) TYPES::QUADRATURE_SCHEME_TYPE::NUMBER_OF_GAUSS

The number of gauss points for the quadrature scheme.

Definition at line 89 of file types.f90.

7.270.2.6 TYPE(QUADRATURE_TYPE),pointer TYPES::QUADRATURE_SCHEME_TYPE::QUADRATURE

The pointer back to the quadrature for a particular quadrature scheme.

Definition at line 88 of file types.f90.

7.271 TYPES::QUADRATURE_TYPE Struct Reference

Contains information on the quadrature to be used for integrating a basis.

Collaboration diagram for TYPES::QUADRATURE_TYPE:

Public Attributes

- INTEGER(INTG) **TYPE**

The type of the quadrature.

- TYPE(BASIS_TYPE), pointer **BASIS**

The pointer back to the basis.

- INTEGER(INTG), allocatable **NUMBER_OF_GAUSS_XI**

NUMBER_OF_GAUSS_XI(ni). For standard Gauss schemes the number of Gauss points to be used in the ni 'th xi direction.

- INTEGER(INTG) **GAUSS_ORDER**

For simplex Gauss schemes the order of the Quadrature scheme i.e., the order/dimension of the polynomial that can be integrated.

- TYPE(QUADRATURE_SCHEME_PTR_TYPE), allocatable **QUADRATURE_SCHEME_MAP**

QUADRATURE_SCHEME_MAP(scheme_idx). The pointer map to the defined quadrature schemes. The size of array is given by BASIS_ROUTINES::BASIS_NUMBER_OF_QUADRATURE_SCHEME_TYPES. If the quadrature scheme is not defined for the particular type then the array element is NULL.

- INTEGER(INTG) **NUMBER_OF_SCHEMES**

The number of quadrature schemes defined for this quadrature.

- TYPE(QUADRATURE_SCHEME_PTR_TYPE), pointer **SCHEMES**

SCHEMES(scheme_idx). The array of pointers to the quadrature schemes defined for the basis. scheme_idx must be between 1 and QUADRATURE_TYPE::NUMBER_OF_SCHEMES.

7.271.1 Detailed Description

Contains information on the quadrature to be used for integrating a basis.

Definition at line 101 of file types.f90.

7.271.2 Member Data Documentation

7.271.2.1 TYPE(BASIS_TYPE),pointer TYPES::QUADRATURE_TYPE::BASIS

The pointer back to the basis.

Definition at line 103 of file types.f90.

7.271.2.2 INTEGER(INTG) TYPES::QUADRATURE_TYPE::GAUSS_ORDER

For simplex Gauss schemes the order of the Quadrature scheme i.e., the order/dimension of the polynomial that can be integrated.

Definition at line 105 of file types.f90.

7.271.2.3 INTEGER(INTG),allocatable TYPES::QUADRATURE_TYPE::NUMBER_OF_- GAUSS_XI

NUMBER_OF_GAUSS_XI(ni). For standard Gauss schemes the number of Gauss points to be used in the ni'th xi direction.

Definition at line 104 of file types.f90.

7.271.2.4 INTEGER(INTG) TYPES::QUADRATURE_TYPE::NUMBER_OF_SCHEMES

The number of quadrature schemes defined for this quadrature.

Definition at line 107 of file types.f90.

7.271.2.5 TYPE(QUADRATURE_SCHEME_PTR_TYPE),allocatable TYPES::QUADRATURE_TYPE::QUADRATURE_SCHEME_MAP

QUADRATURE_SCHEME_MAP(scheme_idx). The pointer map to the defined quadrature schemes. The size of array is given by BASIS_ROUTINES::BASIS_NUMBER_OF_QUADRATURE_SCHEME_TYPES. If the quadrature scheme is not defined for the particular type then the array element is NULL.

See also:

BASIS_ROUTINES_QuadratureSchemes.

Definition at line 106 of file types.f90.

7.271.2.6 TYPE(QUADRATURE_SCHEME_PTR_TYPE),pointer TYPES::QUADRATURE_- TYPE::SCHEMES

SCHEMES(scheme_idx). The array of pointers to the quadrature schemes defined for the basis. scheme_idx must be between 1 and QUADRATURE_TYPE::NUMBER_OF_SCHEMES.

Definition at line 108 of file types.f90.

7.271.2.7 INTEGER(INTG) TYPES::QUADRATURE_TYPE::TYPE

The type of the quadrature.

See also:

BASIS_ROUTINES_QuadratureTypes

Definition at line 102 of file types.f90.

7.272 TYPES::REGION_PTR_TYPE Struct Reference

A buffer type to allow for an array of pointers to a REGION_TYPE.

Collaboration diagram for TYPES::REGION_PTR_TYPE:

Public Attributes

- TYPE(REGION_TYPE), pointer PTR

The pointer to the region.

7.272.1 Detailed Description

A buffer type to allow for an array of pointers to a REGION_TYPE.

Definition at line 1523 of file types.f90.

7.272.2 Member Data Documentation

7.272.2.1 TYPE(REGION_TYPE),pointer TYPES::REGION_PTR_TYPE::PTR

The pointer to the region.

Definition at line 1524 of file types.f90.

7.273 TYPES::REGION_TYPE Struct Reference

Contains information for a region.

Collaboration diagram for TYPES::REGION_TYPE:

Public Attributes

- INTEGER(INTG) **USER_NUMBER**

The user defined identifier for the region. The user number must be unique.

- LOGICAL **REGION_FINISHED**

Is .TRUE. if the region has finished being created, .FALSE. if not.

- TYPE(**VARYING_STRING**) **LABEL**

A user defined label for the region.

- TYPE(**COORDINATE_SYSTEM_TYPE**), pointer **COORDINATE_SYSTEM**

A pointer to the coordinate system used by the region.

- TYPE(**NODES_TYPE**), pointer **NODES**

A pointer to the nodes defined on the region.

- TYPE(**MESHES_TYPE**), pointer **MESHES**

A pointer to the meshes defined on the region.

- TYPE(**FIELDS_TYPE**), pointer **FIELDS**

A pointer to the fields defined on the region.

- TYPE(**EQUATIONS_SETS_TYPE**), pointer **EQUATIONS_SETS**

A pointer to the equation sets defined on the region.

- TYPE(**REGION_TYPE**), pointer **PARENT_REGION**

A pointer to the parent region for the region. If the region has no parent region then it is the global (world) region and PARENT_REGION is NULL.

- INTEGER(INTG) **NUMBER_OF_SUB_REGIONS**

The number of sub-regions defined for the region.

- TYPE(**REGION_PTR_TYPE**), pointer **SUB_REGIONS**

An array of pointers to the sub-regions defined on the region.

7.273.1 Detailed Description

Contains information for a region.

Definition at line 1528 of file types.f90.

7.273.2 Member Data Documentation

7.273.2.1 TYPE(COORDINATE_SYSTEM_TYPE),pointer TYPES::REGION_TYPE::COORDINATE_SYSTEM

A pointer to the coordinate system used by the region.

Definition at line 1532 of file types.f90.

7.273.2.2 TYPE(EQUATIONS_SETS_TYPE),pointer TYPES::REGION_TYPE::EQUATIONS_SETS

A pointer to the equation sets defined on the region.

Definition at line 1536 of file types.f90.

7.273.2.3 TYPE(FIELDS_TYPE),pointer TYPES::REGION_TYPE::FIELDS

A pointer to the fields defined on the region.

Definition at line 1535 of file types.f90.

7.273.2.4 TYPE(VARYING_STRING) TYPES::REGION_TYPE::LABEL

A user defined label for the region.

Definition at line 1531 of file types.f90.

7.273.2.5 TYPE(MESHES_TYPE),pointer TYPES::REGION_TYPE::MESHES

A pointer to the meshes defined on the region.

Definition at line 1534 of file types.f90.

7.273.2.6 TYPE(NODES_TYPE),pointer TYPES::REGION_TYPE::NODES

A pointer to the nodes defined on the region.

Definition at line 1533 of file types.f90.

7.273.2.7 INTEGER(INTG) TYPES::REGION_TYPE::NUMBER_OF_SUB_REGIONS

The number of sub-regions defined for the region.

Definition at line 1538 of file types.f90.

7.273.2.8 TYPE(REGION_TYPE),pointer TYPES::REGION_TYPE::PARENT_REGION

A pointer to the parent region for the region. If the region has no parent region then it is the global (world) region and PARENT_REGION is NULL.

Definition at line 1537 of file types.f90.

7.273.2.9 LOGICAL TYPES::REGION_TYPE::REGION_FINISHED

Is .TRUE. if the region has finished being created, .FALSE. if not.

Definition at line 1530 of file types.f90.

7.273.2.10 TYPE(REGION_PTR_TYPE),pointer TYPES::REGION_TYPE::SUB_REGIONS

An array of pointers to the sub-regions defined on the region.

Definition at line 1539 of file types.f90.

7.273.2.11 INTEGER(INTG) TYPES::REGION_TYPE::USER_NUMBER

The user defined identifier for the region. The user number must be unique.

Definition at line 1529 of file types.f90.

7.274 TYPES::SOLUTION_MAPPING_CREATE_VALUES_- CACHE_TYPE Struct Reference

Contains information about the cached create values for a solution mapping.

Public Attributes

- INTEGER(INTG), allocatable [MATRIX_VARIABLE_TYPES](#)

MATRIX_VARIABLE_TYPES(0:..,equations_set_idx,matrix_idx). The list of matrix variable types in the equations_set_idx'th equations set for the matrix_idx'th solver matrix. *MATRIX_VARIABLE_TYPES(0,equations_set_idx,matrix_idx)* is the number of variable types in the equations_set_idx'th equations set mapped to the matrix_idx'th solver matrix and *MATRIX_VARIABLE_TYPES(1..,equations_set_idx,matrix_idx)* is the list of the variable types in the equations set.

7.274.1 Detailed Description

Contains information about the cached create values for a solution mapping.

Definition at line 1417 of file types.f90.

7.274.2 Member Data Documentation

7.274.2.1 INTEGER(INTG),allocatable TYPES::SOLUTION_MAPPING_CREATE_VALUES_- CACHE_TYPE::MATRIX_VARIABLE_TYPES

MATRIX_VARIABLE_TYPES(0:..,equations_set_idx,matrix_idx). The list of matrix variable types in the equations_set_idx'th equations set for the matrix_idx'th solver matrix. *MATRIX_VARIABLE_TYPES(0,equations_set_idx,matrix_idx)* is the number of variable types in the equations_set_idx'th equations set mapped to the matrix_idx'th solver matrix and *MATRIX_VARIABLE_TYPES(1..,equations_set_idx,matrix_idx)* is the list of the variable types in the equations set.

Definition at line 1418 of file types.f90.

7.275 TYPES::SOLUTION_MAPPING_TYPE Struct Reference

Contains information on the solution mapping between the global equation sets and the solver matrices.

Collaboration diagram for TYPES::SOLUTION_MAPPING_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer SOLUTION
A pointer to the solution for this mapping.
- LOGICAL SOLUTION_MAPPING_FINISHED
Is .TRUE. if the solution mapping has finished being created, .FALSE. if not.
- INTEGER(INTG) NUMBER_OF_ROWS
The number of (local) rows in the solver matrices.
- INTEGER(INTG) TOTAL_NUMBER_OF_ROWS
The total number of rows in the solver matrices.
- INTEGER(INTG) NUMBER_OF_SOLVER_MATRICES
The number of solution matrices in this mapping.
- INTEGER(INTG) NUMBER_OF_EQUATIONS_SETS
The number of equations sets in the solution mapping.
- TYPE(EQUATIONS_SET_PTR_TYPE), allocatable EQUATIONS_SETS
The list of equations sets that are in this solution mapping.
- TYPE(EQUATIONS_SET_TO_SOLVER_MAP_TYPE), allocatable EQUATIONS_SET_TO_-
SOLVER_MAP
EQUATIONS_SET_TO_SOLVER_MAP(equations_set_idx). The mapping from the equations_set_idx'th equations set to the solver matrices.
- TYPE(SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE), allocatable SOLVER_COL_TO_-
EQUATIONS_SETS_MAP
SOLVER_TO_EQUATIONS_SET_MAP(solver_matrix_idx). The mapping from the solver_matrix_idx'th solver matrix to the equations set.
- TYPE(SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE), allocatable SOLVER_ROW_TO_-
EQUATIONS_SET_MAPS
SOLVER_ROW_TO_EQUATIONS_SET_MAPS(row_idx). The mappings from the row_idx'th solver row to the equations set rows.
- TYPE(DOMAIN_MAPPING_TYPE), pointer ROW_DOFS_MAPPING
The domain mapping for the solver rows.
- TYPE(SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE), pointer CREATE_-
VALUES_CACHE
The create values cache for the solution mapping.

7.275.1 Detailed Description

Contains information on the solution mapping between the global equation sets and the solver matrices.
Definition at line 1422 of file types.f90.

7.275.2 Member Data Documentation

7.275.2.1 **TYPE(SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE),pointer TYPES::SOLUTION_MAPPING_TYPE::CREATE_VALUES_CACHE**

The create values cache for the solution mapping.
Definition at line 1434 of file types.f90.

7.275.2.2 **TYPE(EQUATIONS_SET_TO_SOLVER_MAP_TYPE),allocatable TYPES::SOLUTION_MAPPING_TYPE::EQUATIONS_SET_TO_SOLVER_MAP**

EQUATIONS_SET_TO_SOLVER_MAP(equations_set_idx). The mapping from the equations_set_-idx'the equations set to the solver matrices.

Definition at line 1430 of file types.f90.

7.275.2.3 **TYPE(EQUATIONS_SET_PTR_TYPE),allocatable TYPES::SOLUTION_MAPPING_- TYPE::EQUATIONS_SETS**

The list of equations sets that are in this solution mapping.
Definition at line 1429 of file types.f90.

7.275.2.4 **INTEGER(INTG) TYPES::SOLUTION_MAPPING_TYPE::NUMBER_OF_- EQUATIONS_SETS**

The number of equations sets in the solution mapping.
Definition at line 1428 of file types.f90.

7.275.2.5 **INTEGER(INTG) TYPES::SOLUTION_MAPPING_TYPE::NUMBER_OF_ROWS**

The number of (local) rows in the solver matrices.
Definition at line 1425 of file types.f90.

7.275.2.6 **INTEGER(INTG) TYPES::SOLUTION_MAPPING_TYPE::NUMBER_OF_- SOLVER_MATRICES**

The number of solution matrices in this mapping.
Definition at line 1427 of file types.f90.

**7.275.2.7 TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::SOLUTION_MAPPING_-
TYPE::ROW_DOFS_MAPPING**

The domain mapping for the solver rows.

Definition at line 1433 of file types.f90.

**7.275.2.8 TYPE(SOLUTION_TYPE),pointer TYPES::SOLUTION_MAPPING_-
TYPE::SOLUTION**

A pointer to the solution for this mapping.

Definition at line 1423 of file types.f90.

**7.275.2.9 LOGICAL TYPES::SOLUTION_MAPPING_TYPE::SOLUTION_MAPPING_-
FINISHED**

Is .TRUE. if the solution mapping has finished being created, .FALSE. if not.

Definition at line 1424 of file types.f90.

**7.275.2.10 TYPE(SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE),allocatable
TYPES::SOLUTION_MAPPING_TYPE::SOLVER_COL_TO_EQUATIONS_SETS_-
MAP**

SOLVER_TO_EQUATIONS_SET_MAP(solver_matrix_idx). The mapping from the solver_matrix_-idx'th solver matrix to the equations set.

Definition at line 1431 of file types.f90.

**7.275.2.11 TYPE(SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE),allocatable
TYPES::SOLUTION_MAPPING_TYPE::SOLVER_ROW_TO_EQUATIONS_SET_-
MAPS**

SOLVER_ROW_TO_EQUATIONS_SET_MAPS(row_idx). The mappings from the row_idx'th solver row to the equations set rows.

Definition at line 1432 of file types.f90.

**7.275.2.12 INTEGER(INTG) TYPES::SOLUTION_MAPPING_TYPE::TOTAL_NUMBER_-
OF_ROWS**

The total number of rows in the solver matrices.

Definition at line 1426 of file types.f90.

7.276 TYPES::SOLUTION_PTR_TYPE Struct Reference

Collaboration diagram for TYPES::SOLUTION_PTR_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer PTR

7.276.1 Detailed Description

Definition at line 1457 of file types.f90.

7.276.2 Member Data Documentation

7.276.2.1 TYPE(SOLUTION_TYPE),pointer TYPES::SOLUTION_PTR_TYPE::PTR

Definition at line 1458 of file types.f90.

7.277 TYPES::SOLUTION_TYPE Struct Reference

Contains information on the solution of a problem.

Collaboration diagram for TYPES::SOLUTION_TYPE:

Public Attributes

- INTEGER(INTG) [SOLUTION_NUMBER](#)
The number of the solution.
- TYPE([PROBLEM_TYPE](#)), pointer [PROBLEM](#)
A pointer back to the problem for this solution.
- LOGICAL [SOLUTION_FINISHED](#)
Is .TRUE. if the problem solution has finished being created, .FALSE. if not.
- TYPE([EQUATIONS_SET_TYPE](#)), pointer [EQUATIONS_SET_TO_ADD](#)
The next equations set to add to the solution.
- INTEGER(INTG) [EQUATIONS_SET_ADDED_INDEX](#)
The index of the last successfully added equations set.
- TYPE([SOLUTION_MAPPING_TYPE](#)), pointer [SOLUTION_MAPPING](#)
A pointer to the solution mapping for the solution.
- TYPE([SOLVER_TYPE](#)), pointer [SOLVER](#)
A pointer to the solver for the problem.

7.277.1 Detailed Description

Contains information on the solution of a problem.

Definition at line 1444 of file types.f90.

7.277.2 Member Data Documentation

7.277.2.1 INTEGER(INTG) TYPES::SOLUTION_TYPE::EQUATIONS_SET_ADDED_INDEX

The index of the last successfully added equations set.

Definition at line 1452 of file types.f90.

7.277.2.2 TYPE(EQUATIONS_SET_TYPE),pointer TYPES::SOLUTION_TYPE::EQUATIONS_SET_TO_ADD

The next equations set to add to the solution.

Definition at line 1451 of file types.f90.

7.277.2.3 TYPE(PROBLEM_TYPE),pointer TYPES::SOLUTION_TYPE::PROBLEM

A pointer back to the problem for this solution.

Definition at line 1446 of file types.f90.

7.277.2.4 LOGICAL TYPES::SOLUTION_TYPE::SOLUTION_FINISHED

Is .TRUE. if the problem solution has finished being created, .FALSE. if not.

Definition at line 1447 of file types.f90.

7.277.2.5 TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::SOLUTION_TYPE::SOLUTION_MAPPING

A pointer to the solution mapping for the solution.

Definition at line 1453 of file types.f90.

7.277.2.6 INTEGER(INTG) TYPES::SOLUTION_TYPE::SOLUTION_NUMBER

The number of the solution.

Definition at line 1445 of file types.f90.

7.277.2.7 TYPE(SOLVER_TYPE),pointer TYPES::SOLUTION_TYPE::SOLVER

A pointer to the solver for the problem.

Definition at line 1454 of file types.f90.

7.278 TYPES::SOLVER_COL_TO_EQUATIONS_MAP_TYPE Struct Reference

Contains information about the mapping from a solver matrix column to equations matrices and variables.

Public Attributes

- INTEGER(INTG) [NUMBER_OF_EQUATIONS_MATRICES](#)
The number of equations matrices the solver column is mapped to in this equations set.
- INTEGER(INTG), allocatable [EQUATIONS_MATRIX_NUMBERS](#)
EQUATIONS_MATRIX_NUMBERS(i). The i'th equations matrix number in the equations that the solver column is mapped to.
- INTEGER(INTG), allocatable [EQUATIONS_COL_NUMBERS](#)
EQUATIONS_COL_NUMBERS(i). The i'th equations column number in the equation set the solver column is mapped to.
- REAL(DP), allocatable [COUPLING_COEFFICIENTS](#)
COUPLING_COEFFICIENTS(i). The i'th coupling coefficient for solver column mapping.

7.278.1 Detailed Description

Contains information about the mapping from a solver matrix column to equations matrices and variables.
 Definition at line 1371 of file types.f90.

7.278.2 Member Data Documentation

7.278.2.1 REAL(DP),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_MAP_- TYPE::COUPLING_COEFFICIENTS

COUPLING_COEFFICIENTS(i). The i'th coupling coefficient for solver column mapping.
 Definition at line 1375 of file types.f90.

7.278.2.2 INTEGER(INTG),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_MAP_- TYPE::EQUATIONS_COL_NUMBERS

EQUATIONS_COL_NUMBERS(i). The i'th equations column number in the equation set the solver column is mapped to.
 Definition at line 1374 of file types.f90.

7.278.2.3 INTEGER(INTG),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_MAP_- TYPE::EQUATIONS_MATRIX_NUMBERS

EQUATIONS_MATRIX_NUMBERS(i). The i'th equations matrix number in the equations that the solver column is mapped to.

Definition at line 1373 of file types.f90.

**7.278.2.4 INTEGER(INTG) TYPES::SOLVER_COL_TO_EQUATIONS_MAP_-
TYPE::NUMBER_OF_EQUATIONS_MATRICES**

The number of equations matrices the solver column is mapped to in this equations set.

Definition at line 1372 of file types.f90.

7.279 TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_- TYPE Struct Reference

Contains information about the mappings from a solver matrix to the equations in an equations set.

Collaboration diagram for TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE:

Public Attributes

- TYPE(EQUATIONS_TYPE), pointer EQUATIONS
- TYPE(SOLVER_COL_TO_EQUATIONS_MAP_TYPE), allocatable SOLVER_COL_TO_EQUATIONS_MAPS

SOLVER_COL_TO_EQUATIONS_MAPS(col_idx). The mappings from the col_idx'th column of the solver matrix to the equations in the equations set.

7.279.1 Detailed Description

Contains information about the mappings from a solver matrix to the equations in an equations set.

Definition at line 1389 of file types.f90.

7.279.2 Member Data Documentation

7.279.2.1 TYPE(EQUATIONS_TYPE),pointer TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE::EQUATIONS

Definition at line 1390 of file types.f90.

7.279.2.2 TYPE(SOLVER_COL_TO_EQUATIONS_MAP_TYPE),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE::SOLVER_COL_TO_EQUATIONS_MAPS

SOLVER_COL_TO_EQUATIONS_MAPS(col_idx). The mappings from the col_idx'th column of the solver matrix to the equations in the equations set.

Definition at line 1391 of file types.f90.

7.280 TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_-TYPE Struct Reference

Contains information on the mappings from a solver matrix to equations sets.

Collaboration diagram for TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE:

Public Attributes

- INTEGER(INTG) **SOLVER_MATRIX_NUMBER**
The number of this solver matrix.
- TYPE(SOLUTION_MAPPING_TYPE), pointer **SOLUTION_MAPPING**
A pointer to the solution mapping for this solver matrix mapping.
- TYPE(SOLVER_MATRIX_TYPE), pointer **SOLVER_MATRIX**
A pointer to the solver matrix being mapped.
- INTEGER(INTG) **NUMBER_OF_COLUMNS**
The number of columns in this distributed solver matrix.
- TYPE(SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE), allocatable **SOLVER_COL_TO_EQUATIONS_SET_MAPS**
SOLVER_TO_EQUATIONS_SET_MAP(equations_set_idx). The solver columns to equations matrix maps for the col_idx'th column set.
- INTEGER(INTG) **NUMBER_OF_DOFS**
The number of (local) dofs in the solver vector associated with this solver matrix.
- INTEGER(INTG) **TOTAL_NUMBER_OF_DOFS**
The total number of (global) dofs in the solver vector associated with this solver matrix.
- TYPE(SOLVER_COL_TO_VARIABLE_MAP_TYPE), allocatable **SOLVER_COL_TO_VARIABLE_MAPS**
SOLVER_COL_TO_EQUATIONS_MAPS(col_idx). The mappings from the col_idx'th column of the solver matrix to the field variables in the equations set.
- TYPE(DOMAIN_MAPPING_TYPE), pointer **COLUMN_DOFS_MAPPING**
The domain mapping for solver matrix column dofs.

7.280.1 Detailed Description

Contains information on the mappings from a solver matrix to equations sets.

Definition at line 1395 of file types.f90.

7.280.2 Member Data Documentation

7.280.2.1 **TYPE(DOMAIN_MAPPING_TYPE),pointer TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::COLUMN_DOFS_MAPPING**

The domain mapping for solver matrix column dofs.

Definition at line 1405 of file types.f90.

7.280.2.2 **INTEGER(INTG) TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::NUMBER_OF_COLUMNS**

The number of columns in this distributed solver matrix.

Definition at line 1399 of file types.f90.

7.280.2.3 **INTEGER(INTG) TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::NUMBER_OF_DOFS**

The number of (local) dofs in the solver vector associated with this solver matrix.

Definition at line 1401 of file types.f90.

7.280.2.4 **TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::SOLUTION_MAPPING**

A pointer to the solution mapping for this solver matrix mapping.

Definition at line 1397 of file types.f90.

7.280.2.5 **TYPE(SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::SOLVER_COL_TO_EQUATIONS_SET_MAPS**

SOLVER_TO_EQUATIONS_SET_MAP(equations_set_idx). The solver columns to equations matrix maps for the col_idx'th column set.

Definition at line 1400 of file types.f90.

7.280.2.6 **TYPE(SOLVER_COL_TO_VARIABLE_MAP_TYPE),allocatable TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::SOLVER_COL_TO_VARIABLE_MAPS**

SOLVER_COL_TO_EQUATIONS_MAPS(col_idx). The mappings from the col_idx'th column of the solver matrix to the field variables in the equations set.

Definition at line 1404 of file types.f90.

7.280.2.7 **TYPE(SOLVER_MATRIX_TYPE),pointer TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE::SOLVER_MATRIX**

A pointer to the solver matrix being mapped.

Definition at line 1398 of file types.f90.

**7.280.2.8 INTEGER(INTG) TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_-
TYPE::SOLVER_MATRIX_NUMBER**

The number of this solver matrix.

Definition at line 1396 of file types.f90.

**7.280.2.9 INTEGER(INTG) TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_-
TYPE::TOTAL_NUMBER_OF_DOFS**

The total number of (global) dofs in the solver vector associated with this solver matrix.

Definition at line 1402 of file types.f90.

7.281 TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE

Struct Reference

Contains information about mapping the solver column (solver dof) to the field variable dofs in the equations set.

Collaboration diagram for TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE:

Public Attributes

- INTEGER(INTG) [NUMBER_OF_EQUATIONS_SETS](#)

The number of equations sets this column is mapped to.

- INTEGER(INTG), allocatable [EQUATIONS_SET_INDICES](#)

EQUATIONS_SET_INDICES(i). The equations set index of the i'th equations set that this solver column is mapped to.

- TYPE([FIELD_VARIABLE_PTR_TYPE](#)), allocatable [VARIABLE](#)

VARIABLE(i).VARIABLE(i)PTR is a pointer to the field variable that the column is mapped to in the i'th equation set.

- INTEGER(INTG), allocatable [VARIABLE_DOF](#)

VARIABLE_DOF(i). The variable dof number that the column is mapped to in the i'th equations set.

- REAL(DP), allocatable [VARIABLE_COEFFICIENT](#)

VARIABLE_COEFFICIENT(i). The multiplicative coefficient for the mapping to the i'th equations set.

- REAL(DP), allocatable [ADDITIVE_CONSTANT](#)

ADDITIVE_CONSTANT(i). The additive constant for the mapping to the i'th equations set.

7.281.1 Detailed Description

Contains information about mapping the solver column (solver dof) to the field variable dofs in the equations set.

Definition at line 1379 of file types.f90.

7.281.2 Member Data Documentation

7.281.2.1 REAL(DP),allocatable TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE::ADDITIVE_CONSTANT

ADDITIVE_CONSTANT(i). The additive constant for the mapping to the i'th equations set.

Definition at line 1385 of file types.f90.

**7.281.2.2 INTEGER(INTG),allocatable TYPES::SOLVER_COL_TO_VARIABLE_MAP_-
TYPE::EQUATIONS_SET_INDICES**

EQUATIONS_SET_INDICES(i). The equations set index of the i'th equations set that this solver column is mapped to.

Definition at line 1381 of file types.f90.

**7.281.2.3 INTEGER(INTG) TYPES::SOLVER_COL_TO_VARIABLE_MAP_-
TYPE::NUMBER_OF_EQUATIONS_SETS**

The number of equations sets this column is mapped to.

Definition at line 1380 of file types.f90.

**7.281.2.4 TYPE(FIELD_VARIABLE_PTR_TYPE),allocatable TYPES::SOLVER_COL_TO_-
VARIABLE_MAP_TYPE::VARIABLE**

VARIABLE(i).VARIABLE(i)PTR is a pointer to the field variable that the column is mapped to in the i'th equation set.

Definition at line 1382 of file types.f90.

**7.281.2.5 REAL(DP),allocatable TYPES::SOLVER_COL_TO_VARIABLE_MAP_-
TYPE::VARIABLE_COEFFICIENT**

VARIABLE_COEFFICIENT(i). The mulitplicative coefficient for the mapping to the i'th equations set.

Definition at line 1384 of file types.f90.

**7.281.2.6 INTEGER(INTG),allocatable TYPES::SOLVER_COL_TO_VARIABLE_MAP_-
TYPE::VARIABLE_DOF**

VARIABLE_DOF(i). The variable dof number that the column is mapped to in the i'th equations set.

Definition at line 1383 of file types.f90.

7.282 TYPES::SOLVER_MATRICES_TYPE Struct Reference

Contains information on the solver matrices and rhs vector.

Collaboration diagram for TYPES::SOLVER_MATRICES_TYPE:

Public Attributes

- TYPE(SOLVER_TYPE), pointer SOLVER
A pointer to the problem solver.
- LOGICAL SOLVER_MATRICES_FINISHED
Is .TRUE. if the solver matrices have finished being created, .FALSE. if not.
- TYPE(SOLUTION_MAPPING_TYPE), pointer SOLUTION_MAPPING
A pointer to the solution mapping for these solver matrices.
- INTEGER(INTG) NUMBER_OF_ROWS
The number of rows in the distributed solution matrix for this computational node.
- INTEGER(INTG) TOTAL_NUMBER_OF_ROWS
The total number of rows in the distributed solution matrix.
- INTEGER(INTG) LIBRARY_TYPE
The library type for the solver matrices.
- INTEGER(INTG) NUMBER_OF_MATRICES
The number of solver matrices defined for the problem.
- TYPE(SOLVER_MATRIX_PTR_TYPE), allocatable MATRICES
MATRICES(matrix_idx) contains the information on the matrix_idx'th solver matrix.
- LOGICAL UPDATE_RHS_VECTOR
Is .TRUE. if the RHS vector is to be updated.
- TYPE(DISTRIBUTED_VECTOR_TYPE), pointer RHS_VECTOR

7.282.1 Detailed Description

Contains information on the solver matrices and rhs vector.

Definition at line 1221 of file types.f90.

7.282.2 Member Data Documentation

7.282.2.1 INTEGER(INTG) TYPES::SOLVER_MATRICES_TYPE::LIBRARY_TYPE

The library type for the solver matrices.

Definition at line 1227 of file types.f90.

**7.282.2.2 TYPE(SOLVER_MATRIX_PTR_TYPE),allocatable TYPES::SOLVER_MATRICES_-
TYPE::MATRICES**

MATRICES(matrix_idx) contains the information on the matrix_idx'th solver matrix.

Definition at line 1229 of file types.f90.

7.282.2.3 INTEGER(INTG) TYPES::SOLVER_MATRICES_TYPE::NUMBER_OF_MATRICES

The number of solver matrices defined for the problem.

Definition at line 1228 of file types.f90.

7.282.2.4 INTEGER(INTG) TYPES::SOLVER_MATRICES_TYPE::NUMBER_OF_ROWS

The number of rows in the distributed solution matrix for this computational node.

Definition at line 1225 of file types.f90.

**7.282.2.5 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::SOLVER_MATRICES_-
TYPE::RHS_VECTOR**

Definition at line 1231 of file types.f90.

**7.282.2.6 TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::SOLVER_MATRICES_-
TYPE::SOLUTION_MAPPING**

A pointer to the solution mapping for these solver matrices.

Definition at line 1224 of file types.f90.

7.282.2.7 TYPE(SOLVER_TYPE),pointer TYPES::SOLVER_MATRICES_TYPE::SOLVER

A pointer to the problem solver.

Definition at line 1222 of file types.f90.

7.282.2.8 LOGICAL TYPES::SOLVER_MATRICES_TYPE::SOLVER_MATRICES_FINISHED

Is .TRUE. if the solver matrices have finished being created, .FALSE. if not.

Definition at line 1223 of file types.f90.

**7.282.2.9 INTEGER(INTG) TYPES::SOLVER_MATRICES_TYPE::TOTAL_NUMBER_OF_-
ROWS**

The total number of rows in the distributed solution matrix.

Definition at line 1226 of file types.f90.

7.282.2.10 LOGICAL TYPES::SOLVER_MATRICES_TYPE::UPDATE_RHS_VECTOR

Is .TRUE. if the RHS vector is to be updated.

Definition at line 1230 of file types.f90.

7.283 TYPES::SOLVER_MATRIX_PTR_TYPE Struct Reference

Collaboration diagram for TYPES::SOLVER_MATRIX_PTR_TYPE:

Public Attributes

- TYPE(SOLVER_MATRIX_TYPE), pointer PTR

7.283.1 Detailed Description

Definition at line 1216 of file types.f90.

7.283.2 Member Data Documentation

7.283.2.1 TYPE(SOLVER_MATRIX_TYPE),pointer TYPES::SOLVER_MATRIX_PTR_- TYPE::PTR

Definition at line 1217 of file types.f90.

7.284 TYPES::SOLVER_MATRIX_TYPE Struct Reference

Contains information on the solver matrix.

Collaboration diagram for TYPES::SOLVER_MATRIX_TYPE:

Public Attributes

- INTEGER(INTG) **MATRIX_NUMBER**
The number of the solver matrix.
- TYPE(SOLVER_MATRICES_TYPE), pointer **SOLVER_MATRICES**
A pointer to the solver matrices for this solver matrix.
- LOGICAL **UPDATE_MATRIX**
Is .TRUE. if the solver matrix is to be updated.
- INTEGER(INTG) **STORAGE_TYPE**
The storage type for the solver matrix.
- INTEGER(INTG) **NUMBER_OF_COLUMNS**
The number of columns in the distributed solver matrix.
- TYPE(DISTRIBUTED_VECTOR_TYPE), pointer **SOLVER_VECTOR**
A pointer to the distributed solver vector associated with the matrix.
- TYPE(DISTRIBUTED_MATRIX_TYPE), pointer **MATRIX**
A pointer to the distributed solver matrix data.

7.284.1 Detailed Description

Contains information on the solver matrix.

Definition at line 1206 of file types.f90.

7.284.2 Member Data Documentation

7.284.2.1 TYPE(DISTRIBUTED_MATRIX_TYPE),pointer TYPES::SOLVER_MATRIX_- TYPE::MATRIX

A pointer to the distributed solver matrix data.

Definition at line 1213 of file types.f90.

7.284.2.2 INTEGER(INTG) TYPES::SOLVER_MATRIX_TYPE::MATRIX_NUMBER

The number of the solver matrix.

Definition at line 1207 of file types.f90.

7.284.2.3 INTEGER(INTG) TYPES::SOLVER_MATRIX_TYPE::NUMBER_OF_COLUMNS

The number of columns in the distributed solver matrix.

Definition at line 1211 of file types.f90.

**7.284.2.4 TYPE(SOLVER_MATRICES_TYPE),pointer TYPES::SOLVER_MATRIX_-
TYPE::SOLVER_MATRICES**

A pointer to the solver matrices for this solver matrix.

Definition at line 1208 of file types.f90.

**7.284.2.5 TYPE(DISTRIBUTED_VECTOR_TYPE),pointer TYPES::SOLVER_MATRIX_-
TYPE::SOLVER_VECTOR**

A pointer to the distributed solver vector associated with the matrix.

Definition at line 1212 of file types.f90.

7.284.2.6 INTEGER(INTG) TYPES::SOLVER_MATRIX_TYPE::STORAGE_TYPE

The storage type for the solver matrix.

Definition at line 1210 of file types.f90.

7.284.2.7 LOGICAL TYPES::SOLVER_MATRIX_TYPE::UPDATE_MATRIX

Is .TRUE. if the solver matrix is to be updated.

Definition at line 1209 of file types.f90.

7.285 TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_- TYPE Struct Reference

Contains information on the mappings from a solver row to the equations rows.

Public Attributes

- INTEGER(INTG) **NUMBER_OF_ROWS**
The number of rows the solver row is mapped to.
- INTEGER(INTG), allocatable **EQUATIONS_SET**
EQUATIONS_SET(i). The equation set index of i'th row that the solver row is mapped to.
- INTEGER(INTG), allocatable **EQUATIONS_ROW_NUMBER**
EQUATIONS_ROW_NUMBER(i). The i'th equations row number in the equation set the solver row is mapped to.
- REAL(DP), allocatable **COUPLING_COEFFICIENTS**
COUPLING_COEFFICIENTS(i). The i'th coupling coefficient for solver row mapping.

7.285.1 Detailed Description

Contains information on the mappings from a solver row to the equations rows.

Definition at line 1409 of file types.f90.

7.285.2 Member Data Documentation

7.285.2.1 REAL(DP),allocatable TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_- TYPE::COUPLING_COEFFICIENTS

COUPLING_COEFFICIENTS(i). The i'th coupling coefficient for solver row mapping.

Definition at line 1413 of file types.f90.

7.285.2.2 INTEGER(INTG),allocatable TYPES::SOLVER_ROW_TO_EQUATIONS_SET_- MAP_TYPE::EQUATIONS_ROW_NUMBER

EQUATIONS_ROW_NUMBER(i). The i'th equations row number in the equation set the solver row is mapped to.

Definition at line 1412 of file types.f90.

7.285.2.3 INTEGER(INTG),allocatable TYPES::SOLVER_ROW_TO_EQUATIONS_SET_- MAP_TYPE::EQUATIONS_SET

EQUATIONS_SET(i). The equation set index of i'th row that the solver row is mapped to.

Definition at line 1411 of file types.f90.

**7.285.2.4 INTEGER(INTG) TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_-
TYPE::NUMBER_OF_ROWS**

The number of rows the solver row is mapped to.

Definition at line 1410 of file types.f90.

7.286 TYPES::SOLVER_TYPE Struct Reference

Contains information on the type of solver to be used.

Collaboration diagram for TYPES::SOLVER_TYPE:

Public Attributes

- TYPE(SOLUTION_TYPE), pointer SOLUTION
A pointer to the problem solution.
- LOGICAL SOLVER_FINISHED
Is .TRUE. if the problem solver has finished being created, .FALSE. if not.
- TYPE(SOLUTION_MAPPING_TYPE), pointer SOLUTION_MAPPING
A pointer to the problem solution.
- INTEGER(INTG) SOLVE_TYPE
The type of the problem solver.
- INTEGER(INTG) OUTPUT_TYPE
The type of output required.
- INTEGER(INTG) SPARSITY_TYPE
The type of sparsity to use in the solver matrices.
- TYPE(LINEAR_SOLVER_TYPE), pointer LINEAR_SOLVER
A pointer to the linear solver information.
- TYPE(NONLINEAR_SOLVER_TYPE), pointer NONLINEAR_SOLVER
A pointer to the nonlinear solver information.
- TYPE(TIME_INTEGRATION_SOLVER_TYPE), pointer TIME_INTEGRATION_SOLVER
A pointer to the time integration solver information.
- TYPE(EIGENPROBLEM_SOLVER_TYPE), pointer EIGENPROBLEM_SOLVER
A pointer to the eigenproblem solver information.
- TYPE(SOLVER_MATRICES_TYPE), pointer SOLVER_MATRICES
A pointer to the solver matrices for the problem.

7.286.1 Detailed Description

Contains information on the type of solver to be used.

Definition at line 1288 of file types.f90.

7.286.2 Member Data Documentation

7.286.2.1 TYPE(EIGENPROBLEM_SOLVER_TYPE),pointer TYPES::SOLVER_TYPE::EIGENPROBLEM_SOLVER

A pointer to the eigenproblem solver information.

Definition at line 1298 of file types.f90.

7.286.2.2 TYPE(LINEAR_SOLVER_TYPE),pointer TYPES::SOLVER_TYPE::LINEAR_SOLVER

A pointer to the linear solver information.

Definition at line 1295 of file types.f90.

7.286.2.3 TYPE(NONLINEAR_SOLVER_TYPE),pointer TYPES::SOLVER_TYPE::NONLINEAR_SOLVER

A pointer to the nonlinear solver information.

Definition at line 1296 of file types.f90.

7.286.2.4 INTEGER(INTG) TYPES::SOLVER_TYPE::OUTPUT_TYPE

The type of output required.

See also:

[SOLVER_ROUTINES::OutputTypes](#),[SOLVER_ROUTINES](#)

Definition at line 1293 of file types.f90.

7.286.2.5 TYPE(SOLUTION_TYPE),pointer TYPES::SOLVER_TYPE::SOLUTION

A pointer to the problem solution.

Definition at line 1289 of file types.f90.

7.286.2.6 TYPE(SOLUTION_MAPPING_TYPE),pointer TYPES::SOLVER_TYPE::SOLUTION_MAPPING

A pointer to the problem solution.

Definition at line 1291 of file types.f90.

7.286.2.7 INTEGER(INTG) TYPES::SOLVER_TYPE::SOLVE_TYPE

The type of the problem solver.

See also:

[SOLVER_ROUTINES::SolverTypes](#),[SOLVER_ROUTINES](#)

Definition at line 1292 of file types.f90.

7.286.2.8 LOGICAL TYPES::SOLVER_TYPE::SOLVER_FINISHED

Is .TRUE. if the problem solver has finished being created, .FALSE. if not.

Definition at line 1290 of file types.f90.

7.286.2.9 TYPE(SOLVER_MATRICES_TYPE),pointer TYPES::SOLVER_TYPE::SOLVER_MATRICES

A pointer to the solver matrices for the problem.

Definition at line 1299 of file types.f90.

7.286.2.10 INTEGER(INTG) TYPES::SOLVER_TYPE::SPARSITY_TYPE

The type of sparsity to use in the solver matrices.

See also:

SOLVER_ROUTINES_SparsityTypes,[SOLVER_ROUTINES](#)

Definition at line 1294 of file types.f90.

7.286.2.11 TYPE(TIME_INTEGRATION_SOLVER_TYPE),pointer TYPES::SOLVER_TYPE::TIME_INTEGRATION_SOLVER

A pointer to the time integration solver information.

Definition at line 1297 of file types.f90.

7.287 TYPES::SOURCE_EQUATIONS_MATRICES_MAP_TYPE Struct Reference

Public Attributes

- INTEGER(INTG), allocatable [EQUATIONS_ROW_TO_SOURCE_DOF_MAP](#)
EQUATIONS_ROW_TO_SOURCE_DOF_MAP(row_idx). The mapping from the row_idx'th row of the equations to the source dof.
- INTEGER(INTG), allocatable [SOURCE_DOF_TO_EQUATIONS_ROW_MAP](#)
SOURCE_DOF_TO_EQUATIONS_ROW_MAP(source_dof_idx). The mapping from the source_dof_idx'th source dof to the equations row.

7.287.1 Detailed Description

Definition at line 1009 of file types.f90.

7.287.2 Member Data Documentation

7.287.2.1 INTEGER(INTG),allocatable TYPES::SOURCE_EQUATIONS_MATRICES_MAP_- TYPE::EQUATIONS_ROW_TO_SOURCE_DOF_MAP

EQUATIONS_ROW_TO_SOURCE_DOF_MAP(row_idx). The mapping from the row_idx'th row of the equations to the source dof.

Definition at line 1010 of file types.f90.

7.287.2.2 INTEGER(INTG),allocatable TYPES::SOURCE_EQUATIONS_MATRICES_MAP_- TYPE::SOURCE_DOF_TO_EQUATIONS_ROW_MAP

SOURCE_DOF_TO_EQUATIONS_ROW_MAP(source_dof_idx). The mapping from the source_dof_idx'th source dof to the equations row.

Definition at line 1011 of file types.f90.

7.288 TYPES::TIME_INTEGRATION_SOLVER_TYPE Struct Reference

Contains information for a time integration solver.

Collaboration diagram for TYPES::TIME_INTEGRATION_SOLVER_TYPE:

Public Attributes

- TYPE([SOLVER_TYPE](#)), pointer **SOLVER**
A pointer to the problem_solver.
- INTEGER(INTG) **SOLVER_LIBRARY**
The library type for the time integration solver.

7.288.1 Detailed Description

Contains information for a time integration solver.

Definition at line 1276 of file types.f90.

7.288.2 Member Data Documentation

7.288.2.1 TYPE(SOLVER_TYPE),pointer TYPES::TIME_INTEGRATION_SOLVER_- TYPE::SOLVER

A pointer to the problem_solver.

Definition at line 1277 of file types.f90.

7.288.2.2 INTEGER(INTG) TYPES::TIME_INTEGRATION_SOLVER_TYPE::SOLVER_- LIBRARY

The library type for the time integration solver.

See also:

[SOLVER_ROUTINES::SolverLibraries](#),[SOLVER_ROUTINES](#)

Definition at line 1278 of file types.f90.

7.289 TYPES::VARIABLE_TO_EQUATIONS_COLUMN_MAP_- TYPE Struct Reference

Contains the information about the mapping of a variable DOF to an equations matrix column.

Public Attributes

- INTEGER(INTG), allocatable **COLUMN_DOF**

COLUMN_DOF(dof_idx). The equations column number for this equations matrix that the dof_idx'th variable DOF is mapped to.

7.289.1 Detailed Description

Contains the information about the mapping of a variable DOF to an equations matrix column.

Definition at line 994 of file types.f90.

7.289.2 Member Data Documentation

7.289.2.1 INTEGER(INTG),allocatable TYPES::VARIABLE_TO_EQUATIONS_COLUMN_- MAP_TYPE::COLUMN_DOF

COLUMN_DOF(dof_idx). The equations column number for this equations matrix that the dof_idx'th variable DOF is mapped to.

Definition at line 995 of file types.f90.

7.290 TYPES::VARIABLE_TO_EQUATIONS_MATRICES_- MAP_TYPE Struct Reference

Contains the mapping for a dependent variable type to the equations matrices.

Collaboration diagram for TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE:

Public Attributes

- INTEGER(INTG) [VARIABLE_INDEX](#)
The variable index for this variable to equations matrices map.
- INTEGER(INTG) [VARIABLE_TYPE](#)
The variable type for this variable to equations matrices map.
- TYPE([FIELD_VARIABLE_TYPE](#)), pointer [VARIABLE](#)
A pointer to the field variable for this variable to equations matrices map.
- INTEGER(INTG) [NUMBER_OF_EQUATIONS_MATRICES](#)
The number of equations matrices this variable type is mapped to. If the number is -1 the variable is mapped to the RHS vector. If the number is zero then this variable type is not involved in the equations set and the rest of the type is not allocated.
- INTEGER(INTG), allocatable [EQUATIONS_MATRIX_NUMBERS](#)
EQUATIONS_MATRIX_NUMBERS(i). The equations matrix number for the i'th matrix that this variable type is mapped to.
- TYPE([VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE](#)), allocatable [DOF_TO_COLUMNS_MAPS](#)
DOF_TO_COLUMNS_MAPS(i). The variable dof to equations columns for the i'th equations matrix.
- INTEGER(INTG), allocatable [DOF_TO_ROWS_MAP](#)
DOF_TO_ROWS_MAP(dof_idx). The row number that the dof_idx'th variable dof is mapped to.

7.290.1 Detailed Description

Contains the mapping for a dependent variable type to the equations matrices.

Definition at line 999 of file types.f90.

7.290.2 Member Data Documentation

7.290.2.1 TYPE([VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE](#)),allocatable TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE::DOF_TO_COLUMNS_MAPS

DOF_TO_COLUMNS_MAPS(i). The variable dof to equations columns for the i'th equations matrix.

Definition at line 1005 of file types.f90.

**7.290.2.2 INTEGER(INTG),allocatable TYPES::VARIABLE_TO_EQUATIONS_MATRICES_-
MAP_TYPE::DOF_TO_ROWS_MAP**

DOF_TO_ROWS_MAP(dof_idx). The row number that the dof_idx'th variable dof is mapped to.

Definition at line 1006 of file types.f90.

**7.290.2.3 INTEGER(INTG),allocatable TYPES::VARIABLE_TO_EQUATIONS_MATRICES_-
MAP_TYPE::EQUATIONS_MATRIX_NUMBERS**

EQUATIONS_MATRIX_NUMBERS(i). The equations matrix number for the i'th matrix that this variable type is mapped to.

Definition at line 1004 of file types.f90.

**7.290.2.4 INTEGER(INTG) TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_-
TYPE::NUMBER_OF_EQUATIONS_MATRICES**

The number of equations matrices this variable type is mapped to. If the number is -1 the variable is mapped to the RHS vector. If the number is zero then this variable type is not involved in the equations set and the rest of the type is not allocated.

Definition at line 1003 of file types.f90.

**7.290.2.5 TYPE(FIELD_VARIABLE_TYPE),pointer TYPES::VARIABLE_TO_EQUATIONS_-
MATRICES_MAP_TYPE::VARIABLE**

A pointer to the field variable for this variable to equations matrices map.

Definition at line 1002 of file types.f90.

**7.290.2.6 INTEGER(INTG) TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_-
TYPE::VARIABLE_INDEX**

The variable index for this variable to equations matrices map.

Definition at line 1000 of file types.f90.

**7.290.2.7 INTEGER(INTG) TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_-
TYPE::VARIABLE_TYPE**

The variable type for this variable to equations matrices map.

Definition at line 1001 of file types.f90.

7.291 TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE Struct Reference

Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).

Public Attributes

- INTEGER(INTG), allocatable [COLUMN_NUMBERS](#)

COLUMN_NUMBERS(variable_dof_idx). The solver column number (solver dof) that the variable_dof_idx'th variable dof is mapped to.

- REAL(DP), allocatable [COUPLING_COEFFICIENTS](#)

COUPLING_COEFFICIENTS(variable_dof_idx). The multiplicative constant for the mapping between the variable_dof_idx'th variable dof and the solver dof.

- REAL(DP), allocatable [ADDITIVE_CONSTANTS](#)

ADDITIVE_CONSTANTS(variable_dof_idx). The additive constant for the mapping between the variable_dof_idx'th variable dof and the solver dof.

7.291.1 Detailed Description

Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).

[Todo](#)

rename solver col to be solver dof here

Definition at line 1328 of file types.f90.

7.291.2 Member Data Documentation

7.291.2.1 REAL(DP),allocatable TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE::ADDITIVE_CONSTANTS

ADDITIVE_CONSTANTS(variable_dof_idx). The additive constant for the mapping between the variable_dof_idx'th variable dof and the solver dof.

Definition at line 1331 of file types.f90.

7.291.2.2 INTEGER(INTG),allocatable TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE::COLUMN_NUMBERS

COLUMN_NUMBERS(variable_dof_idx). The solver column number (solver dof) that the variable_dof_idx'th variable dof is mapped to.

Definition at line 1329 of file types.f90.

**7.291.2.3 REAL(DP),allocatable TYPES::VARIABLE_TO_SOLVER_COL_MAP_-
TYPE::COUPLING_COEFFICIENTS**

COUPLING_COEFFICIENTS(variable_dof_idx). The multiplicative constant for the mapping between the variable_dof_idx'th variable dof and the solver dof.

Definition at line 1330 of file types.f90.

7.292 TYPES::VECTOR_TYPE Struct Reference

Contains information for a vector.

Public Attributes

- INTEGER(INTG) **ID**
The ID of the vector.
- LOGICAL **VECTOR_FINISHED**
Is .TRUE. if the vector has finished being created, .FALSE. if not.
- INTEGER(INTG) **N**
The length of the vector.
- INTEGER(INTG) **DATA_TYPE**
The data type of the vector.
- INTEGER(INTG) **SIZE**
The size of the data array of the vector.
- INTEGER(INTG), allocatable **DATA_INTG**
DATA_INTG(i). The integer data for an integer vector. The i'th component contains the data for the i'th component vector data on the domain.
- REAL(SP), allocatable **DATA_SP**
DATA_SP(i). The real data for a single precision real vector. The i'th component contains the data for the i'th component vector data on the domain.
- REAL(DP), allocatable **DATA_DP**
DATA_DP(i). The real data for a double precision real vector. The i'th component contains the data for the i'th component vector data on the domain.
- LOGICAL, allocatable **DATA_L**
DATA_L(i). The real for a logical vector. The i'th component contains the data for the i'th component vector data on the domain.

7.292.1 Detailed Description

Contains information for a vector.

Definition at line 554 of file types.f90.

7.292.2 Member Data Documentation

7.292.2.1 REAL(DP),allocatable TYPES::VECTOR_TYPE::DATA_DP

DATA_DP(i). The real data for a double precision real vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 562 of file types.f90.

7.292.2.2 INTEGER(INTG),allocatable TYPES::VECTOR_TYPE::DATA_INTG

DATA_INTG(i). The integer data for an integer vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 560 of file types.f90.

7.292.2.3 LOGICAL,allocatable TYPES::VECTOR_TYPE::DATA_L

DATA_L(i). The real for a logical vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 563 of file types.f90.

7.292.2.4 REAL(SP),allocatable TYPES::VECTOR_TYPE::DATA_SP

DATA_SP(i). The real data for a single precision real vector. The i'th component contains the data for the i'th component vector data on the domain.

Definition at line 561 of file types.f90.

7.292.2.5 INTEGER(INTG) TYPES::VECTOR_TYPE::DATA_TYPE

The data type of the vector.

See also:

[MATRIX_VECTOR::DataTypes](#)

Definition at line 558 of file types.f90.

7.292.2.6 INTEGER(INTG) TYPES::VECTOR_TYPE::ID

The ID of the vector.

Definition at line 555 of file types.f90.

7.292.2.7 INTEGER(INTG) TYPES::VECTOR_TYPE::N

The length of the vector.

Definition at line 557 of file types.f90.

7.292.2.8 INTEGER(INTG) TYPES::VECTOR_TYPE::SIZE

The size of the data array of the vector.

Definition at line 559 of file types.f90.

7.292.2.9 LOGICAL TYPES::VECTOR_TYPE::VECTOR_FINISHED

Is .TRUE. if the vector has finished being created, .FALSE. if not.

Definition at line 556 of file types.f90.

Chapter 8

File Documentation

8.1 additional_doc/Installation.dox File Reference

8.2 additional_doc/Test.dox File Reference

8.3 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/base_routines.f90 File Reference

Id

[base_routines.f90](#) 4 2007-08-09 17:08:02Z chrisbradley

Classes

- struct [BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE](#)
Contains information for an item in the routine list for diagnostics or timing.
- struct [BASE_ROUTINES::ROUTINE_LIST_TYPE](#)
Contains information for the routine list for diagnostics or timing.
- struct [BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE](#)
Contains information for an item in the routine invocation stack.
- struct [BASE_ROUTINES::ROUTINE_STACK_TYPE](#)
Contains information for the routine invocation stack.
- interface [BASE_ROUTINES::interface](#)
- interface [BASE_ROUTINES::FLAG_ERROR](#)
Flags an error condition.
- interface [BASE_ROUTINES::FLAG_WARNING](#)
Flags a warning to the user.

Namespaces

- namespace [BASE_ROUTINES](#)
This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

Functions

- subroutine [BASE_ROUTINES::ENTERS](#) (NAME, ERR, ERROR,*)
Records the entry into the named procedure and initialises the error code.
- subroutine [BASE_ROUTINES::ERRORS](#) (NAME, ERR, ERROR)
Records the exiting error of the subroutine.
- subroutine [BASE_ROUTINES::EXITS](#) (NAME)
Records the exit out of the named procedure.
- subroutine [BASE_ROUTINES::FLAG_ERROR_C](#) (STRING, ERR, ERROR,*)
Sets the error string specified by a character string and flags an error.

- subroutine **BASE_ROUTINES::FLAG_ERROR_VS** (STRING, ERR, ERROR,*)

Sets the error string specified by a varying string and flags an error.
- subroutine **BASE_ROUTINES::FLAG_WARNING_C** (STRING, ERR, ERROR,*)

Writes a warning message specified by a character string to the user.
- subroutine **BASE_ROUTINES::FLAG_WARNING_VS** (STRING, ERR, ERROR,*)

Writes a warning message specified by a varying string to the user.
- subroutine **BASE_ROUTINES::BASE_ROUTINES_FINALISE** (ERR, ERROR,*)

Finalises the base_routines module and deallocates all memory.
- subroutine **BASE_ROUTINES::BASE_ROUTINES_INITIALISE** (ERR, ERROR,*)

Initialises the variables required for the base_routines module.
- subroutine **BASE_ROUTINES::DIAGNOSTICS_SET_OFF** (ERR, ERROR,*)

Sets diagnostics off.
- subroutine **BASE_ROUTINES::DIAGNOSTICS_SET_ON** (DIAG_TYPE, LEVEL_LIST, DIAG_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets diagnostics on.
- subroutine **BASE_ROUTINES::OUTPUT_SET_OFF** (ERR, ERROR,*)

Sets writes file echo output off.
- subroutine **BASE_ROUTINES::OUTPUT_SET_ON** (ECHO_FILENAME, ERR, ERROR,*)

Sets writes file echo output on.
- subroutine **BASE_ROUTINES::TIMING_SET_OFF** (ERR, ERROR,*)

Sets timing off.
- subroutine **BASE_ROUTINES::TIMING_SET_ON** (TIMING_TYPE, TIMING_SUMMARY_FLAG, TIMING_FILENAME, ROUTINE_LIST, ERR, ERROR,*)

Sets timing on.
- subroutine **BASE_ROUTINES::TIMING_SUMMARY_OUTPUT** (ERR, ERROR,*)

Outputs the timing summary.
- subroutine **BASE_ROUTINES::WRITE_STR** (ID, ERR, ERROR,*)

Writes the output string to a specified output stream.

Variables

- INTEGER(INTG), parameter **BASE_ROUTINES::MAX_OUTPUT_LINES** = 500

Maximum number of lines that can be output.
- INTEGER(INTG), parameter **BASE_ROUTINES::GENERAL_OUTPUT_TYPE** = 1

General output type.

- INTEGER(INTG), parameter **BASE_ROUTINES::DIAGNOSTIC_OUTPUT_TYPE** = 2
Diagnostic output type.
- INTEGER(INTG), parameter **BASE_ROUTINES::TIMING_OUTPUT_TYPE** = 3
Timing output type.
- INTEGER(INTG), parameter **BASE_ROUTINES::ERROR_OUTPUT_TYPE** = 4
Error output type.
- INTEGER(INTG), parameter **BASE_ROUTINES::HELP_OUTPUT_TYPE** = 5
Help output type.
- INTEGER(INTG), parameter **BASE_ROUTINES::ECHO_FILE_UNIT** = 10
File unit for echo files.
- INTEGER(INTG), parameter **BASE_ROUTINES::DIAGNOSTICS_FILE_UNIT** = 11
File unit for diagnostic files.
- INTEGER(INTG), parameter **BASE_ROUTINES::TIMING_FILE_UNIT** = 12
File unit for timing files.
- INTEGER(INTG), parameter **BASE_ROUTINES::LEARN_FILE_UNIT** = 13
File unit for learn files.
- INTEGER(INTG), parameter **BASE_ROUTINES::IO1_FILE_UNIT** = 21
File unit for general IO 1 files.
- INTEGER(INTG), parameter **BASE_ROUTINES::IO2_FILE_UNIT** = 22
File unit for general IO 2 files.
- INTEGER(INTG), parameter **BASE_ROUTINES::IO3_FILE_UNIT** = 23
File unit for general IO 3 files.
- INTEGER(INTG), parameter **BASE_ROUTINES::IO4_FILE_UNIT** = 24
File unit for general IO 4 files.
- INTEGER(INTG), parameter **BASE_ROUTINES::IO5_FILE_UNIT** = 25
File unit for general IO 5 files.
- INTEGER(INTG), parameter **BASE_ROUTINES::TEMPORARY_FILE_UNIT** = 80
File unit for temporary files.
- INTEGER(INTG), parameter **BASE_ROUTINES::OPEN_COMFILE_UNIT** = 90
File unit for open command files.
- INTEGER(INTG), parameter **BASE_ROUTINES::START_READ_COMFILE_UNIT** = 90
First file unit for read command files.

- INTEGER(INTG), parameter **BASE_ROUTINES::STOP_READ_COMFILE_UNIT** = 99
Last file unit for read command files.
- INTEGER(INTG), parameter **BASE_ROUTINES::ALL_DIAG_TYPE** = 1
Type for setting diagnostic output in all routines.
- INTEGER(INTG), parameter **BASE_ROUTINES::IN_DIAG_TYPE** = 2
Type for setting diagnostic output in one routine.
- INTEGER(INTG), parameter **BASE_ROUTINES::FROM_DIAG_TYPE** = 3
Type for setting diagnostic output from one routine downwards.
- INTEGER(INTG), parameter **BASE_ROUTINES::ALL_TIMING_TYPE** = 1
Type for setting timing output in all routines.
- INTEGER(INTG), parameter **BASE_ROUTINES::IN_TIMING_TYPE** = 2
Type for setting timing output in one routine.
- INTEGER(INTG), parameter **BASE_ROUTINES::FROM_TIMING_TYPE** = 3
Type for setting timing output from one routine downwards.
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS**
.TRUE. if diagnostic output is required in any routines.
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS1**
.TRUE. if level 1 diagnostic output is active in the current routine
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS2**
.TRUE. if level 2 diagnostic output is active in the current routine
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS3**
.TRUE. if level 3 diagnostic output is active in the current routine
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS4**
.TRUE. if level 4 diagnostic output is active in the current routine
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS5**
.TRUE. if level 5 diagnostic output is active in the current routine
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS_LEVEL1**
.TRUE. if the user has requested level 1 diagnostic output to be active
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS_LEVEL2**
.TRUE. if the user has requested level 2 diagnostic output to be active
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS_LEVEL3**
.TRUE. if the user has requested level 3 diagnostic output to be active
- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS_LEVEL4**
.TRUE. if the user has requested level 4 diagnostic output to be active

- LOGICAL, save **BASE_ROUTINES::DIAGNOSTICS_LEVEL**
.TRUE. if the user has requested level 5 diagnostic output to be active
- LOGICAL, save **BASE_ROUTINES::DIAG_ALL_SUBROUTINES**
.TRUE. if diagnostic output is required in all routines
- LOGICAL, save **BASE_ROUTINES::DIAG_FROM_SUBROUTINE**
.TRUE. if diagnostic output is required from a particular routine
- LOGICAL, save **BASE_ROUTINES::DIAG_FILE_OPEN**
.TRUE. if the diagnostic output file is open
- LOGICAL, save **BASE_ROUTINES::DIAG_OR_TIMING**
.TRUE. if diagnostics or time is .TRUE.
- LOGICAL, save **BASE_ROUTINES::ECHO_OUTPUT**
.TRUE. if all output is to be echoed to the echo file
- LOGICAL, save **BASE_ROUTINES::TIMING**
.TRUE. if timing output is required in any routines.
- LOGICAL, save **BASE_ROUTINES::TIMING_SUMMARY**
.TRUE. if timing output will be summary form via a TIMING_SUMMARY_OUTPUT call otherwise timing will be output for routines when the routine exits
- LOGICAL, save **BASE_ROUTINES::TIMING_ALL_SUBROUTINES**
.TRUE. if timing output is required in all routines
- LOGICAL, save **BASE_ROUTINES::TIMING_FROM_SUBROUTINE**
.TRUE. if timing output is required from a particular routine
- LOGICAL, save **BASE_ROUTINES::TIMING_FILE_OPEN**
.TRUE. if the timing output file is open
- CHARACTER(LEN=MAXSTRLEN), save **BASE_ROUTINES::OP_STRING**
The array of lines to output.
- TYPE(ROUTINE_LIST_TYPE), save **BASE_ROUTINES::DIAG_ROUTINE_LIST**
The list of routines for which diagnostic output is required.
- TYPE(ROUTINE_LIST_TYPE), save **BASE_ROUTINES::TIMING_ROUTINE_LIST**
The list of routines for which timing output is required.
- TYPE(ROUTINE_STACK_TYPE), save **BASE_ROUTINES::ROUTINE_STACK**
The routime invocation stack.

8.3.1 Detailed Description

Id

[base_routines.f90](#) 4 2007-08-09 17:08:02Z chrispb Bradley

Author:

Chris Bradley This module contains all the low-level base routines e.g., all debug, control, and low-level communication routines.

8.3.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [base_routines.f90](#).

8.4 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/basis_routines.f90 File Reference

8.5 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_c.c File Reference

Include dependency graph for binary_file_c.c:

Defines

- #define INTEGERTYPE 1
- #define SHORTINTTYPE 2
- #define LONGINTTYPE 3
- #define FLOATTYPE 4
- #define DOUBLETYPE 5
- #define QUADRUPLETYP 6
- #define CHARTYPE 7
- #define LOGICALTYPE 8
- #define COMPLEXTYPE 9
- #define DOUBLECOMPLEX 10
- #define QUADRUPLECOMPLEX 11
- #define MAXBINFILES 99
- #define SAMEENDIAN 0
- #define FLIPENDIAN 1

Typedefs

- typedef int logical

Functions

- void BinaryCloseFile (int *fileid, int *err, char *error_string)
- void BinaryOpenFile (int *fileid, char *filename, char *access_code, int *err, char *error_string)
- void BinaryReadFile (int *fileid, int *endian, int *number_of_items, int *item_type, char *data, int *err, char *error_string)
- void BinarySetFile (int *fileid, int *set_code, int *err, char *error_string)
- void BinarySkipFile (int *fileid, int *number_of_bytes, int *err, char *error_string)
- void BinaryWriteFile (int *fileid, int *endian, int *number_of_items, int *item_type, char *data, int *err, char *error_string)
- void IsBinaryFileOpen (int *fileid, int *returncode, int *err, char *error_string)
- void IsEndBinaryFile (int *fileid, int *returncode, int *err, char *error_string)

Variables

- FILE * binaryfiles [MAXBINFILES]

8.5.1 Define Documentation

8.5.1.1 #define CHARTYPE 7

Definition at line 118 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.2 #define COMPLEXTYPE 9

Definition at line 120 of file binary_file_c.c.

8.5.1.3 #define DOUBLECOMPLEXTYPE 10

Definition at line 121 of file binary_file_c.c.

8.5.1.4 #define DOUBLETYPE 5

Definition at line 116 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.5 #define FLIPENDIAN 1

Definition at line 128 of file binary_file_c.c.

8.5.1.6 #define FLOATTYPE 4

Definition at line 115 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.7 #define INTEGERTYPE 1

Definition at line 112 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.8 #define LOGICALTYPE 8

Definition at line 119 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.9 #define LONGINTTYPE 3

Definition at line 114 of file binary_file_c.c.

8.5.1.10 #define MAXBINFILES 99

Definition at line 126 of file binary_file_c.c.

Referenced by BinaryCloseFile(), BinaryOpenFile(), BinaryReadFile(), BinarySetFile(), BinarySkipFile(), BinaryWriteFile(), IsBinaryFileOpen(), and IsEndBinaryFile().

8.5.1.11 #define QUADRUPLECOMPLEXTYPE 11

Definition at line 122 of file binary_file_c.c.

8.5.1.12 #define QUADRUPLETTYPE 6

Definition at line 117 of file binary_file_c.c.

8.5.1.13 #define SAMEENDIAN 0

Definition at line 127 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.1.14 #define SHORTINTTYPE 2

Definition at line 113 of file binary_file_c.c.

Referenced by BinaryReadFile(), and BinaryWriteFile().

8.5.2 Typedef Documentation**8.5.2.1 typedef int logical**

Definition at line 132 of file binary_file_c.c.

8.5.3 Function Documentation**8.5.3.1 void BinaryCloseFile (int *fileid, int *err, char *error_string)**

Definition at line 195 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.3.2 void BinaryOpenFile (int *fileid, char *filename, char *access_code, int *err, char *error_string)

Definition at line 227 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.3.3 void BinaryReadFile (int *fileid, int *endian, int *number_of_items, int *item_type, char *data, int *err, char *error_string)

Definition at line 265 of file binary_file_c.c.

References binaryfiles, CHARTYPE, DOUBLETYPE, FLOATTYPE, INTEGERTYPE, LOGICALTYPE, MAXBINFILES, SAMEENDIAN, and SHORTINTTYPE.

8.5.3.4 void BinarySetFile (int *fileid, int *set_code, int *err, char *error_string)

Definition at line 375 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.3.5 void BinarySkipFile (int *fileid, int *number_of_bytes, int *err, char *error_string)

Definition at line 440 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.3.6 void BinaryWriteFile (int *fileid, int *endian, int *number_of_items, int *item_type, char *data, int *err, char *error_string)

Definition at line 483 of file binary_file_c.c.

References binaryfiles, CHARTYPE, DOUBLETYPE, FLOATTYPE, INTEGERTYPE, LOGICALTYPE, MAXBINFILES, SAMEENDIAN, and SHORTINTTYPE.

8.5.3.7 void IsBinaryFileOpen (int *fileid, int *returncode, int *err, char *error_string)

Definition at line 587 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.3.8 void IsEndBinaryFile (int *fileid, int *returncode, int *err, char *error_string)

Definition at line 621 of file binary_file_c.c.

References binaryfiles, and MAXBINFILES.

8.5.4 Variable Documentation

8.5.4.1 FILE* binaryfiles[MAXBINFILES]**Initial value:**

```
{NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL}
```

Definition at line 180 of file binary_file_c.c.

Referenced by BinaryCloseFile(), BinaryOpenFile(), BinaryReadFile(), BinarySetFile(), BinarySkipFile(), BinaryWriteFile(), IsBinaryFileOpen(), and IsEndBinaryFile().

8.6 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/binary_file_f.f90 File Reference

Id

[binary_file_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- struct [BINARY_FILE::BINARY_FILE_INFO_TYPE](#)
- struct [BINARY_FILE::BINARY_FILE_TYPE](#)
- struct [BINARY_FILE::BINARY_TAG_TYPE](#)
- interface [BINARY_FILE::interface](#)
- interface [BINARY_FILE::READ_BINARY_FILE](#)
- interface [BINARY_FILE::WRITE_BINARY_FILE](#)

Namespaces

- namespace [BINARY_FILE](#)

This module handles the reading and writing of binary files.

Functions

- LOGICAL [BINARY_FILE::INQUIRE_OPEN_BINARY_FILE](#) (FILEID)
- LOGICAL [BINARY_FILE::INQUIRE_EOF_BINARY_FILE](#) (FILEID, ERR, ERROR)
- subroutine [BINARY_FILE::CLOSE_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [BINARY_FILE::CLOSE_CMISS_BINARY_FILE](#) (FILEID, ERR, ERROR,*)
- subroutine [BINARY_FILE::OPEN_BINARY_FILE](#) (FILEID, COMMAND, FILENAME, ERR, ERROR,*)
- subroutine [BINARY_FILE::OPEN_CMISS_BINARY_FILE](#) (FILEID, FILE_TYPE, NUMBER_TA_S,&VERSION, FILEVERSION, COMMAND, EXTENSION, FILENAME, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_INTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_INTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_SINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_SINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_LINTG](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_LINTG1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_SP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_SP1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::READ_BINARY_FILE_DP](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

- subroutine `BINARY_FILE::READ_BINARY_FILE_DP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_CHARACTER` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_LOGICAL1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_SPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_SPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_DPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_FILE_DPC1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::READ_BINARY_TAG_HEADER` (FILEID, TAG, ERR, ERROR,*)
- subroutine `BINARY_FILE::RESET_BINARY_NUMBER_TAGS` (FILEID, NUMBER_TAGS, ERR, ERROR,*)
- subroutine `BINARY_FILE::SET_BINARY_FILE` (FILEID, SET_CODE, ERR, ERROR,*)
- subroutine `BINARY_FILE::SKIP_CM_BINARY_HEADER` (FILEID, SKIP, ERR, ERROR,*)
- subroutine `BINARY_FILE::SKIP_BINARY_FILE` (FILEID, NUMBER_BYTES, ERR, ERROR,*)
- subroutine `BINARY_FILE::SKIP_BINARY_TAGS` (FILEID, TAG, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_INTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LINTG1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_DP1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_CHARACTER` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_LOGICAL1` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine `BINARY_FILE::WRITE_BINARY_FILE_SPC` (FILEID, NUM_DATA, DATA, ERR, ERROR,*)

- subroutine [BINARY_FILE::WRITE_BINARY_FILE_SPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::WRITE_BINARY_FILE_DPC](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::WRITE_BINARY_FILE_DPC1](#) (FILEID, NUM_DATA, DATA, ERR, ERROR,*)
- subroutine [BINARY_FILE::WRITE_BINARY_TAG_HEADER](#) (FILEID, TAG, ERR, ERROR,*)

Variables

- INTEGER(INTG), dimension, parameter [BINARY_FILE::MAX_NUM_BINARY_FILES](#) = 99
- INTEGER(INTG), parameter [BINARY_FILE::FILE_BEGINNING](#) = 0
- INTEGER(INTG), parameter [BINARY_FILE::FILE_CURRENT](#) = 1
- INTEGER(INTG), parameter [BINARY_FILE::FILE_END](#) = 2
- INTEGER(INTG), parameter [BINARY_FILE::FILE_SAME_ENDIAN](#) = 0
- INTEGER(INTG), parameter [BINARY_FILE::FILE_CHANGE_ENDIAN](#) = 1
- INTEGER(INTG), parameter [BINARY_FILE::BINARY_FILE_READABLE](#) = 1
- INTEGER(INTG), parameter [BINARY_FILE::BINARY_FILE_WRITABLE](#) = 2
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_IDENTITY](#) = 7
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_MATRIX_FILE](#) = 1
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_HISTORY_FILE](#) = 2
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_SIGNAL_FILE](#) = 3
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_IDENTITY_HEADER](#) = 1
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_MACHINE_HEADER](#) = 2
- INTEGER(INTG), parameter [BINARY_FILE::CMISS_BINARY_FILE_HEADER](#) = 3
- LOGICAL, save [BINARY_FILE::BINARY_FILE_USED](#) = .FALSE.

8.6.1 Detailed Description

Id

[binary_file_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module handles the reading and writing of binary files.

Todo

Fix naming convention and update to current code standard.

8.6.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [binary_file_f.f90](#).

8.7 d:/Users/tyu011/workspace/OpenCMISS-trunk/srcblas.f90 File Reference

Id

[blas.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [BLAS::interface](#)

Namespaces

- namespace [BLAS](#)

This module contains the interface descriptions to the BLAS routines.

8.7.1 Detailed Description

Id

[blas.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains the interface descriptions to the [BLAS](#) routines.

8.7.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [blas.f90](#).

8.8 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/classical_field_routines.f90 File Reference

Id

[classical_field_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [CLASSICAL_FIELD_ROUTINES](#)

This module handles all classical field class routines.

Functions

- subroutine [CLASSICAL_FIELD_ROUTINES::CL](#) (EQUATIONS_SET, EQUATIONS_TYPE, EQUATIONS_SUBTYPE,&ERR, ERROR,*)

Sets/changes the problem type and subtype for a classical field equation set class.

- subroutine [CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SETFINITE_ELEMENT_CALCULATE](#) (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matrices and rhs vector for the given element number for a classical field class finite element equation set.

- subroutine [CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_EQUATIONS_SET_SETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the equations set for a classical field equations set class.

- subroutine [CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_CLASS_TYPE_SET](#) (PROBLEM, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem type and subtype for a classical field problem class.

- subroutine [CLASSICAL_FIELD_ROUTINES::CLASSICAL_FIELD_PROBLEM_SETUP](#) (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the problem for a classical field problem class.

8.8.1 Detailed Description

Id

[classical_field_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all classical field routines.

8.8.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [classical_field_routines.f90](#).

8.9 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss.f90 File Reference

Id

[cmiss.f90](#) 20 2007-05-28 20:22:52Z cpb

Namespaces

- namespace [CMISS](#)
The top level cmiss module.

Functions

- subroutine [CMISS::CMISS_FINALISE](#) (ERR, ERROR,*)
Finalises [CMISS](#).
- subroutine [CMISS::CMISS_INITIALISE](#) (ERR, ERROR,*)
Initialises [CMISS](#).
- subroutine [CMISS::CMISS_WRITE_ERROR](#) (ERR, ERROR)
Writes the error string to screen.

Variables

- INTEGER(INTG), parameter [CMISS::CMISS_MAJOR_VERSION](#) = 0
- INTEGER(INTG), parameter [CMISS::CMISS_MINOR_VERSION](#) = 1
- INTEGER(INTG), parameter [CMISS::CMISS_BUILD_VERSION](#) = 1

8.9.1 Detailed Description

Id

[cmiss.f90](#) 20 2007-05-28 20:22:52Z cpb

Author:

Chris Bradley The top level cmiss module.

Definition in file [cmiss.f90](#).

8.10 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_mpi.f90 File Reference

Id

cmiss_mpi.f90 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [CMISS_MPI](#)

This module contains CMISS MPI routines.

Functions

- subroutine [CMISS_MPI::MPI_ERROR_CHECK](#) (ROUTINE, MPI_ERR_CODE, ERR, ERROR,*)

Checks to see if an MPI error has occurred during an MPI call and flags a CMISS error if it has.

8.10.1 Detailed Description

Id

cmiss_mpi.f90 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains CMISS MPI routines.

8.10.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss_mpi.f90](#).

8.11 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_parmetis.f90 File Reference

Id

[cmiss_parmetis.f90](#) 9 2007-05-15 13:52:02Z cpb

Classes

- interface [CMISS_PARMETIS::interface](#)

Namespaces

- namespace [CMISS_PARMETIS](#)

This module is a CMISS buffer module to the ParMETIS library.

Functions

- subroutine [CMISS_PARMETIS::PARMETIS_PARTKWAY](#) (VERTEX_DISTANCE, XADJ, ADJNCY, VERTEX_WEIGHT, ADJ_WEIGHT, WEIGHT_FLA, NUM_FLAG, NCON,&NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartKway routine.

- subroutine [CMISS_PARMETIS::PARMETIS_PARTMESHKWAY](#) (ELEMENT_DISTANCE, ELEMENT_PTR, ELEMENT_INDEX, ELEMENT_WEIGHT, WEIGHT_FLAG, NUM_FLAG, CON,&NUMBER_COMMON_NODES, NUMBER_PARTS, TP_WEIGHTS, UB_VEC, OPTIONS, NUMBER_EDGES_CUT, PARTITION, COMMUNICATOR, ERR, ERROR,*)

Buffer routine to the ParMetis ParMETIS_V3_PartMeshKway routine.

8.11.1 Detailed Description

Id

[cmiss_parmetis.f90](#) 9 2007-05-15 13:52:02Z cpb

Author:

Chris Bradley This module is a CMISS buffer module to the ParMETIS library.

8.11.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss_parmetis.f90](#).

8.12 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_petsc.f90 File Reference

Id

cmiss_petsc.f90 27 2007-07-24 16:52:51Z cpb

Include dependency graph for cmiss_petsc.f90:

Classes

- interface [CMISS_PETSC::interface](#)

Namespaces

- namespace [CMISS_PETSC](#)

This module is a CMISS buffer module to the PETSc library.

Functions

- subroutine [CMISS_PETSC::PETSC_ERRORHANDLING_SET_OFF](#) (ERR, ERROR,*)
Set PETSc error handling on.
- subroutine [CMISS_PETSC::PETSC_ERRORHANDLING_SET_ON](#) (ERR, ERROR,*)
Set PETSc error handling on.
- subroutine [CMISS_PETSC::PETSC_FINALIZE](#) (ERR, ERROR,*)
Buffer routine to the PETSc PetscFinalize routine.
- subroutine [CMISS_PETSC::PETSC_INITIALIZE](#) (FILE, ERR, ERROR,*)
Buffer routine to the PETSc PetscInitialize routine.
- subroutine [CMISS_PETSC::PETSC_ISFINALISE](#) (IS_, ERR, ERROR,*)
- subroutine [CMISS_PETSC::PETSC_ISINITIALISE](#) (IS_, ERR, ERROR,*)
- subroutine [CMISS_PETSC::PETSC_ISDESTROY](#) (IS_, ERR, ERROR,*)
Buffer routine to the PETSc ISDestroy routine.
- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGFINALISE](#) (ISLOCALTOGLOBALMAPPING_, ERR, ERROR,*)
- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE](#) (ISLOCALTOGLOBALMAPPING_, ERR, ERROR,*)
- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLY](#) (CTX, TYPE, NIN, IDXIN, NOUT, IDXOUT, ERR, ERROR,*)
Buffer routine to the PETSc ISLocalToGlobalMappingApply routine.
- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS](#) (CTX, ISIN, ISOUT, ERR, ERROR,*)
Buffer routine to the PETSc ISLocalToGlobalMappingApplyIS routine.

- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGCREATE](#) (COMMUNICATOR, N, GLOBALNUM, CTX, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingCreate routine.
- subroutine [CMISS_PETSC::PETSC_ISLOCALTOGLOBALMAPPINGDESTROY](#) (CTX, ERR, ERROR,*)

Buffer routine to the PETSc ISLocalToGlobalMappingDestroy routine.
- subroutine [CMISS_PETSC::PETSC_KSPCREATE](#) (COMMUNICATOR, KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPCreate routine.
- subroutine [CMISS_PETSC::PETSC_KSPDESTROY](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPDestroy routine.
- subroutine [CMISS_PETSC::PETSC_KSPGETCONVERGEDREASON](#) (KSP_, REASON, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetConvergedReason routine.
- subroutine [CMISS_PETSC::PETSC_KSPGETITERATIONNUMBER](#) (KSP_, ITERATION_NUMBER, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetIterationNumber routine.
- subroutine [CMISS_PETSC::PETSC_KSPGETPC](#) (KSP_, PC_, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetPC routine.
- subroutine [CMISS_PETSC::PETSC_KSPGETRESIDUALNORM](#) (KSP_, RESIDUAL_NORM, ERR, ERROR,*)

Buffer routine to the PETSc KSPGetResidualNorm routine.
- subroutine [CMISS_PETSC::PETSC_KSPFINALISE](#) (KSP_, ERR, ERROR,*)
 • subroutine [CMISS_PETSC::PETSC_KSPINITIALISE](#) (KSP_, ERR, ERROR,*)
 • subroutine [CMISS_PETSC::PETSC_KSPSETFROMOPTIONS](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetFromOptions routine.
- subroutine [CMISS_PETSC::PETSC_KSPSETOPERATORS](#) (KSP_, AMAT, PMAT, FLAG, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetOperators routine.
- subroutine [CMISS_PETSC::PETSC_KSPSETTOLERANCES](#) (KSP_, RTOL, ATOL, DTOL, MAXITS, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetTolerances routine.
- subroutine [CMISS_PETSC::PETSC_KSPSETTYPE](#) (KSP_, METHOD, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetType routine.
- subroutine [CMISS_PETSC::PETSC_KSPSETUP](#) (KSP_, ERR, ERROR,*)

Buffer routine to the PETSc KSPSetUp routine.

- subroutine [CMISS_PETSC::PETSC_KSPSOLVE](#) (KSP_, B, X, ERR, ERROR,*)

Buffer routine to the PETSc KSPSolve routine.
- subroutine [CMISS_PETSC::PETSC_LOGPRINTSUMMARY](#) (COMMUNICATOR, FILE, ERR, ERROR,*)

Buffer routine to the PETSc PetscLogPrintSummary routine.
- subroutine [CMISS_PETSC::PETSC_MATASSEMBLYBEGIN](#) (A, ASSEMBLY_TYPE, ERR, ERROR,*)

Buffer routine to the PETSc MatAssemblyBegin routine.
- subroutine [CMISS_PETSC::PETSC_MATASSEMBLYEND](#) (A, ASSEMBLY_TYPE, ERR, ERROR,*)

Buffer routine to the PETSc MatAssemblyEnd routine.
- subroutine [CMISS_PETSC::PETSC_MATCREATE](#) (COMMUNICATOR, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreate routine.
- subroutine [CMISS_PETSC::PETSC_MATCREATEMPIAIJ](#) (COMMUNICATOR, LOCAL_M, LOCAL_N, GLOB_L_M, GLOBAL_N, DIAG_NUMBER_NZ_PERROW, DIAG_NUMBER_NZ_EACHROW,&OFFDIAG_NUMBER_NZ_PERROW, OFFDIAG_NUMBER_NZ_EACHROW, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateMPIAIJ routine.
- subroutine [CMISS_PETSC::PETSC_MATCREATEMPIDENSE](#) (COMMUNICATOR, LOCAL_M, LOCAL_N, GLOBAL_M, GLOBAL_N, MATRIX_DATA, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateMPIDense routine.
- subroutine [CMISS_PETSC::PETSC_MATCREATESEQAIJ](#) (COMMUNICATOR, M, N, NUMBER_NZ_PERROW, NUMBER_NZ_EACHROW, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateSeqAIJ routine.
- subroutine [CMISS_PETSC::PETSC_MATCREATESEQDENSE](#) (COMMUNICATOR, M, N, MATRIX_DATA, A, ERR, ERROR,*)

Buffer routine to the PETSc MatCreateSeqDense routine.
- subroutine [CMISS_PETSC::PETSC_MATDESTROY](#) (A, ERR, ERROR,*)

Buffer routine to the PETSc MatDestroy routine.
- subroutine [CMISS_PETSC::PETSC_MATGETARRAY](#) (A, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc MatGetArray routine.
- subroutine [CMISS_PETSC::PETSC_MATGETOWNERSHIPRANGE](#) (A, FIRST_ROW, LAST_ROW, ERR, ERROR,*)

Buffer routine to the PETSc MatGetOwnershipRange routine.
- subroutine [CMISS_PETSC::PETSC_MATGETVALUES](#) (A, M, M_INDICES, N, N_INDICES, VALUES, ERR, ERROR,*)

Buffer routine to the PETSc MatGetValues routine.
- subroutine [CMISS_PETSC::PETSC_MATFINALISE](#) (MAT_, ERR, ERROR,*)

- subroutine [CMISS_PETSC::PETSC_MATINITIALISE](#) (MAT_, ERR, ERROR,*)
 - Buffer routine to the PETSc MatRestoreArray routine.*
- subroutine [CMISS_PETSC::PETSC_MATRESTOREARRAY](#) (A, ERR, ERROR,*)
 - Buffer routine to the PETSc MatRestoreArray routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETLOCALTOGLOBALMAPPING](#) (A, CTX, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetLocalToGlobalMapping routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETOPTION](#) (A, OPTION, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetOption routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETSIZES](#) (A, LOCAL_M, LOCAL_N, GLOBAL_M, GLOBAL_N, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetSizes routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETVALUE](#) (A, ROW, COL, VALUE, INSERT_MODE, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetValue routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETVALUES](#) (A, M, M_INDICES, N, N_INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetValues routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETVALUELOCAL](#) (A, ROW, COL, VALUE, INSERT_MODE, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetValueLocal routine.*
- subroutine [CMISS_PETSC::PETSC_MATSETVALUESLOCAL](#) (A, M, M_INDICES, N, N_INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)
 - Buffer routine to the PETSc MatSetValuesLocal routine.*
- subroutine [CMISS_PETSC::PETSC_MATVIEW](#) (A, V, ERR, ERROR,*)
 - Buffer routine to the PETSc MatView routine.*
- subroutine [CMISS_PETSC::PETSC_MATZEROENTRIES](#) (A, ERR, ERROR,*)
 - Buffer routine to the PETSc MatZeroEntries routine.*
- subroutine [CMISS_PETSC::PETSC_PCFINALISE](#) (PC_, ERR, ERROR,*)
 - Buffer routine to the PETSc PCFinalise routine.*
- subroutine [CMISS_PETSC::PETSC_PCINITIALISE](#) (PC_, ERR, ERROR,*)
 - Buffer routine to the PETSc PCInitialise routine.*
- subroutine [CMISS_PETSC::PETSC_PCSETTYPE](#) (PC_, METHOD, ERR, ERROR,*)
 - Buffer routine to the PETSc PCSetType routine.*
- subroutine [CMISS_PETSC::PETSC_VECFINALISE](#) (VEC_, ERR, ERROR,*)
 - Buffer routine to the PETSc VecFinalise routine.*
- subroutine [CMISS_PETSC::PETSC_VECINITIALISE](#) (VEC_, ERR, ERROR,*)
 - Buffer routine to the PETSc VecInitialise routine.*
- subroutine [CMISS_PETSC::PETSC_VECASSEMBLYBEGIN](#) (X, ERR, ERROR,*)
 - Buffer routine to the PETSc VecAssemblyBegin routine.*
- subroutine [CMISS_PETSC::PETSC_VECASSEMBLYEND](#) (X, ERR, ERROR,*)
 - Buffer routine to the PETSc VecAssemblyEnd routine.*

- subroutine [CMISS_PETSC::PETSC_VECCREATE](#) (COMMUNICATOR, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreate routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATEGHOST](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, NUMBER_GHOST, GHOSTS, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateGhost routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATEGHOSTWITHARRAY](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, NUMBER_GHOST, GHOSTS, ARRAY, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateGhostWithArray routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATEMPI](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateMPI routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATEMPIWITHARRAY](#) (COMMUNICATOR, LOCAL_SIZE, GLOBAL_SIZE, ARRAY, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateMPIWithArray routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATESEQ](#) (COMMUNICATOR, SIZE, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateSeq routine.
- subroutine [CMISS_PETSC::PETSC_VECCREATESEQWITHARRAY](#) (COMMUNICATOR, SIZE, ARRAY, X, ERR, ERROR,*)
Buffer routine to the PETSc VecCreateSeqWithArray routine.
- subroutine [CMISS_PETSC::PETSC_VECDESTROY](#) (X, ERR, ERROR,*)
Buffer routine to the PETSc VecDestroy routine.
- subroutine [CMISS_PETSC::PETSC_VECDUPLICATE](#) (OLD, NEW, ERR, ERROR,*)
Buffer routine to the PETSc VecDuplicate routine.
- subroutine [CMISS_PETSC::PETSC_VECGETARRAY](#) (X, ARRAY, ERR, ERROR,*)
Buffer routine to the PETSc VecGetArray routine.
- subroutine [CMISS_PETSC::PETSC_VECGETARRAYF90](#) (X, ARRAY, ERR, ERROR,*)
Buffer routine to the PETSc VecGetArrayF90 routine.
- subroutine [CMISS_PETSC::PETSC_VECGETLOCALSIZE](#) (X, SIZE, ERR, ERROR,*)
Buffer routine to the PETSc VecGetLocalSize routine.
- subroutine [CMISS_PETSC::PETSC_VECGETOWNERSHIPRANGE](#) (X, LOW, HIGH, ERR, ERROR,*)
Buffer routine to the PETSc VecGetOwnershipRange routine.
- subroutine [CMISS_PETSC::PETSC_VECGETSIZE](#) (X, SIZE, ERR, ERROR,*)
Buffer routine to the PETSc VecGetSize routine.
- subroutine [CMISS_PETSC::PETSC_VECGETVALUES](#) (X, N, INDICES, VALUES, ERR, ERROR,*)

Buffer routine to the PETSc VecGetValues routine.

- subroutine [CMISS_PETSC::PETSC_VECGHOSTGETLOCALFORM](#) (G, L, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostGetLocalForm routine.

- subroutine [CMISS_PETSC::PETSC_VECGHOSTRESTORELOCALFORM](#) (G, L, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostRestoreLocalForm routine.

- subroutine [CMISS_PETSC::PETSC_VECGHOSTUPDATEBEGIN](#) (X, INSERT_MODE, SCATTER_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostUpdateBegin routine.

- subroutine [CMISS_PETSC::PETSC_VECGHOSTUPDATEEND](#) (X, INSERT_MODE, SCATTER_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecGhostUpdateEnd routine.

- subroutine [CMISS_PETSC::PETSC_VCRESTOREARRAY](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecRestoreArray routine.

- subroutine [CMISS_PETSC::PETSC_VCRESTOREARRAYF90](#) (X, ARRAY, ERR, ERROR,*)

Buffer routine to the PETSc VecRestoreArrayF90 routine.

- subroutine [CMISS_PETSC::PETSC_VECSET](#) (X, VALUE, ERR, ERROR,*)

Buffer routine to the PETSc VecSet routine.

- subroutine [CMISS_PETSC::PETSC_VECSETFROMOPTIONS](#) (X, ERR, ERROR,*)

Buffer routine to the PETSc VecSetFromOptions routine.

- subroutine [CMISS_PETSC::PETSC_VECSETLOCALTOGLOBALMAPPING](#) (X, CTX, ERR, ERROR,*)

Buffer routine to the PETSc VecSetLocalToGlobalMapping routine.

- subroutine [CMISS_PETSC::PETSC_VECSETVALUES](#) (X, N, INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetValues routine.

- subroutine [CMISS_PETSC::PETSC_VECSETVALUESLOCAL](#) (X, N, INDICES, VALUES, INSERT_MODE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetValuesLocal routine.

- subroutine [CMISS_PETSC::PETSC_VECSETSIZES](#) (X, LOCAL_SIZE, GLOBAL_SIZE, ERR, ERROR,*)

Buffer routine to the PETSc VecSetSizes routine.

- subroutine [CMISS_PETSC::PETSC_VECVIEW](#) (X, V, ERR, ERROR,*)

Buffer routine to the PETSc VecView routine.

Variables

- LOGICAL, save [CMISS_PETSC::PETSC_HANDLE_ERROR](#)

8.12.1 Detailed Description

Id

[cmiss_petsc.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module is a [CMISS](#) buffer module to the PETSc library.

8.12.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss_petsc.f90](#).

8.13 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/cmiss_- petsc_types.f90 File Reference

Id

[cmiss_petsc_types.f90](#) 27 2007-07-24 16:52:51Z cpb

Include dependency graph for cmiss_petsc_types.f90:

Classes

- struct [CMISS_PETSC_TYPES::PETSC_IS_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_ISLOCALTOGLOBALMAPPING_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_KSP_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_MAT_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_PC_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_SNES_TYPE](#)
- struct [CMISS_PETSC_TYPES::PETSC_VEC_TYPE](#)

Namespaces

- namespace [CMISS_PETSC_TYPES](#)

This module contains types related to the PETSc library.

8.13.1 Detailed Description

Id

[cmiss_petsc_types.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains type definitions related to the PETSc library.

8.13.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [cmiss_petsc_types.f90](#).

8.14 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/computational_environment.f90 File Reference

Id

[computational_environment.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- struct [COMP_ENVIRONMENT::CACHE_TYPE](#)
Contains information on a cache hierarchy.
- struct [COMP_ENVIRONMENT::COMPUTATIONAL_NODE_TYPE](#)
Contains information on a computational node containing a number of processors.
- struct [COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE](#)
Contains information on the MPI type to transfer information about a computational node.
- struct [COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_TYPE](#)
Contains information on the computational environment the program is running in.

Namespaces

- namespace [COMP_ENVIRONMENT](#)
This module contains all computational environment variables.

Functions

- subroutine [COMP_ENVIRONMENT::COMPUTATIONAL_NODE_FINALISE](#)
([COMPUTATIONAL_NODE](#), ERR, ERROR,*)
Finalises the computational node data structures and deallocates all memory.
- subroutine [COMP_ENVIRONMENT::COMPUTATIONAL_NODE_INITIALISE](#)
([COMPUTATIONAL_NODE](#), RANK, ERR, ERROR,*)
Initialises the computational node data structures.
- subroutine [COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_FINALISE](#)
(ERR, ERROR,*)
Finalises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).
- subroutine [COMP_ENVIRONMENT::COMPUTATIONAL_NODE_MPI_TYPE_INITIALISE](#)
([COMPUTATIONAL_NODE](#), ERR, ERROR,*)
Initialises the data structure containing the MPI type information for the [COMPUTATIONAL_NODE_TYPE](#).
- subroutine [COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_FINALISE](#) (ERR, ERROR,*)

Finalises the computational environment data structures and deallocates all memory.

- subroutine **COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT_INITIALISE**
(ERR, ERROR,*)

Initialises the computational environment data structures.

- INTEGER(INTG) **COMP_ENVIRONMENT::COMPUTATIONAL_NODE_NUMBER_GET**
(ERR, ERROR)

Returns the number/rank of the computational nodes.

- INTEGER(INTG) **COMP_ENVIRONMENT::COMPUTATIONAL_NODES_NUMBER_GET**
(ERR, ERROR)

Returns the number of computational nodes.

Variables

- TYPE(COMPUTATIONAL_ENVIRONMENT_TYPE) **COMP_ENVIRONMENT::COMPUTATIONAL_ENVIRONMENT**

The computational environment the program is running in.

- TYPE(MPI_COMPUTATIONAL_NODE_TYPE) **COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE_DATA**

The MPI data on the computational nodes.

8.14.1 Detailed Description

Id

[computational_environment.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module contains all computational environment variables.

8.14.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [computational_environment.f90](#).

8.15 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/constants.f90 File Reference

8.16 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/coordinate_routines.f90 File Reference

Id

[coordinate_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- struct [COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE](#)
- struct [COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE](#)
- interface [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC](#)
- interface [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC](#)
- interface [COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET](#)
- interface [COORDINATE_ROUTINES::DXZ](#)
- interface [COORDINATE_ROUTINES::D2ZX](#)
- interface [COORDINATE_ROUTINES::DZX](#)
- interface [COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_CONVERT_TO_RC](#)
- interface [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY](#)

Namespaces

- namespace [COORDINATE_ROUTINES](#)

This module contains all coordinate transformation and support routines.

Functions

- REAL(DP) [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_DP](#)
(COORDINATE_SYSTEM, Z, ERR, ERROR)
- REAL(SP) [COORDINATE_ROUTINES::COORDINATE_CONVERT_FROM_RC_SP](#)
(COORDINATE_SYSTEM, Z, ERR, ERROR)
- REAL(DP) [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_DP](#)
(COORDINATE_SYSTEM, X, ERR, ERROR)
- REAL(SP) [COORDINATE_ROUTINES::COORDINATE_CONVERT_TO_RC_SP](#)
(COORDINATE_SYSTEM, X, ERR, ERROR)
- REAL(DP) [COORDINATE_ROUTINES::COORDINATE_DELTA_CALCULATE_DP](#)
(COORDINATE_SYSTEM, X, Y, ERR, ERROR)
- subroutine [COORDINATE_ROUTINES::COORDINATE_METRICS_CALCULATE](#)
(COORDINATE_SYSTEM, JACOBIAN_TYPE, METRICS, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_NORMAL_CALCULATE](#)
(COORDINATE_SYSTEM, REVERSE, X, N, ERR, ERROR,*)

- INTEGER(INTG) [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_GET](#)(COORDINATE_SYSTEM)
- REAL(DP) [COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_GET](#)(COORDINATE_SYSTEM, ERR, ERROR)
- INTEGER(INTG) [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_GET](#)(COORDINATE_SYSTEM)
- INTEGER(INTG) [COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_GET](#)(COORDINATE_SYSTEM)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_NUMBER](#)(USER_NUMBER, DIMENSION, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET_PTR](#)(COORDINATE_SYSTEM, DIMENSION, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_NUMBER](#)(USER_NUMBER, FOCUS, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET_PTR](#)(COORDINATE_SYSTEM, FOCUS, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER](#)(USER_NUMBER, RADIAL_INTERPOLATION_TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR](#)(COORDINATE_SYSTEM, RADIAL_INTERPOLATION_TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_NUMBER](#)(USER_NUMBER, TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET_PTR](#)(COORDINATE_SYSTEM, TYPE, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_NUMBER](#)(USER_NUMBER, ORIGIN, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET_PTR](#)(COORDINATE_SYSTEM, ORIGIN, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_NUMBER](#)(USER_NUMBER, ORIENTATION, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIENTATION_SET_PTR](#)(COORDINATE_SYSTEM, ORIENTATION, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_START](#)(USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_CREATE_FINISH](#)(COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_NUMBER](#)(USER_NUMBER, ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY_PTR](#)(COORDINATE_SYSTEM, ERR, ERROR,*)
- REAL(DP) [COORDINATE_ROUTINES::DXZ_DP](#)(COORDINATE_SYSTEM, I, X, ERR, ERROR)
- REAL(DP) [COORDINATE_ROUTINES::D2ZX_DP](#)(COORDINATE_SYSTEM, I, J, X, ERR, ERROR)
- REAL(DP) [COORDINATE_ROUTINES::DZX_DP](#)(COORDINATE_SYSTEM, I, X, ERR, ERROR)
- subroutine [COORDINATE_ROUTINES::CO](#)(COORDINATE_SYSTEM, PART_DERIV_TYPE, X, Z,&ERR, ERROR,*)
- subroutine [COORDINATE_ROUTINES::CO](#)(COORDINATE_SYSTEM, PART_DERIV_TYPE, X, Z,&ERR, ERROR,*)

- subroutine **COORDINATE_ROUTINES::COORDINATE_DERIVATIVE_NORM**
(COORDINATE_SYSTEM, PART_DERIV_INDEX, INTERPOLATED_POINT, DERIV_NORM, ERR, ERROR,*)
- subroutine **COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_ADJUST**
(COORDINATE_SYSTEM, PARTIAL_DERIVATIVE_INDEX, VALUE, ERR, ERROR,*)
- subroutine **COORDINATE_ROUTINES::COORDINATE_INTERPOLATION_PARAMETERS_ADJUST**
(COORDINATE_SYSTEM, INTERPOLATION_PARAMETERS, ERR, ERROR,*)
- subroutine **COORDINATE_ROUTINES::COORDINATE_SYSTEM_USER_NUMBER_FIND**
(USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine **COORDINATE_ROUTINES::COORDINATE_SYSTEMS_FINALISE** (ERR,
ERROR,*)
- subroutine **COORDINATE_ROUTINES::COORDINATE_SYSTEMS_INITIALISE** (ERR,
ERROR,*)

Variables

- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_RECTANGULAR_CARTESIAN_TYPE** = 1
Rectangular Cartesian coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_CYCLINDRICAL_POLAR_TYPE** = 2
Cylindrical polar coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_SPHERICAL_POLAR_TYPE** = 3
Spherical polar coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_PROLATE_SPHEROIDAL_TYPE** = 4
Prolate spheroidal coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_OBLATE_SPHEROIDAL_TYPE** = 5
Oblate spheroidal coordinate system type.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_NO_RADIAL_INTERPOLATION_TYPE** = 0
No radial interpolation.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_RADIAL_INTERPOLATION_TYPE** = 1
r radial interpolation
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE** = 2
 r^2 radial interpolation
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE** = 3
 r^3 radial interpolation

- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_JACOBIAN_LINE_TYPE** = 1
Line type Jacobian.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_JACOBIAN_AREA_TYPE** = 2
Area type Jacobian.
- INTEGER(INTG), parameter **COORDINATE_ROUTINES::COORDINATE_JACOBIAN_VOLUME_TYPE** = 3
Volume type Jacobian.
- CHARACTER(LEN=21) **COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE_STRING** = & (/ , & , & , & , & /)
- TYPE(COORDINATE_SYSTEMS_TYPE) **COORDINATE_ROUTINES::COORDINATE_SYSTEMS**
- TYPE(COORDINATE_SYSTEM_TYPE), pointer **COORDINATE_ROUTINES::GLOBAL_COORDINATE_SYSTEM**

8.16.1 Detailed Description

Id

`coordinate_routines.f90` 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all coordinate transformation and support routines.

8.16.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [coordinate_routines.f90](#).

8.17 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/distributed_matrix_vector.f90 File Reference

8.18 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/domain_mappings.f90 File Reference

Id

[domain_mappings.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [DOMAIN_MAPPINGS](#)

This module handles all domain mappings routines.

Functions

- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_FINALISE](#) (ADJACENT_DOMAIN, ERR, ERROR,*)

Finalises the adjacent domain and deallocates all memory for a domain mapping.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_ADJACENT_DOMAIN_INITIALISE](#) (ADJACENT_DOMAIN, ERR, ERROR,*)

Initialise the adjacent domain for a domain mapping.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_LOCAL_FROM_GLOBAL_CALCULATE](#) (DOMAIN_MAPPING, ERR, ERROR,*)

Calculates the domain mappings local map from a domain mappings global map.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_FINALISE](#) (DOMAIN_MAPPING, ERR, ERROR,*)

Finalises the mapping for a domain mappings mapping and deallocates all memory.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_FINALISE](#) (MAPPING_GLOBAL_MAP, ERR, ERROR,*)

Finalises the global mapping in the given domain mappings.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_GLOBAL_INITIALISE](#) (MAPPING_GLOBAL_MAP, ERR, ERROR,*)

Finalises the global mapping in the given domain mappings.
- subroutine [DOMAIN_MAPPINGS::DOMAIN_MAPPINGS_MAPPING_INITIALISE](#) (DOMAIN_MAPPING, NUMBER_OF_DOMAINS, ERR, ERROR,*)

Initialises the mapping for a domain mappings mapping.

Variables

- INTEGER(INTG), parameter [DOMAIN_MAPPINGS::DOMAIN_LOCAL_INTERNAL](#) = 1

The domain item is internal to the domain.

- INTEGER(INTG), parameter **DOMAIN_MAPPINGS::DOMAIN_LOCAL_BOUNDARY** = 2
The domain item is on the boundary of the domain.
- INTEGER(INTG), parameter **DOMAIN_MAPPINGS::DOMAIN_LOCAL_GHOST** = 3
The domain item is ghosted from another domain.

8.18.1 Detailed Description

Id

[domain_mappings.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all domain mappings routines.

8.18.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [domain_mappings.f90](#).

8.19 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/elasticity_routines.f90 File Reference

Id

[elasticity_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [ELASTICITY_ROUTINES](#)

This module handles all elasticity class routines.

8.19.1 Detailed Description

Id

[elasticity_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all elasticity routines.

8.19.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [elasticity_routines.f90](#).

8.20 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/electromechanics_routines.f90 File Reference

Id

electromechanics_routines.f90 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [ELECTROMECHANICS_ROUTINES](#)
This module handles all electromechanics class routines.

8.20.1 Detailed Description

Id

electromechanics_routines.f90 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all electromechanics routines.

8.20.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [electromechanics_routines.f90](#).

8.21 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_- mapping_routines.f90 File Reference**

**8.22 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_-
matrices_routines.f90 File Reference**

8.23 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_constants.f90 File Reference

Id

[equations_set_constants.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace EQUATIONS_SET_CONSTANTS

This module defines all constants shared across equations set routines.

Variables

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_CLASS = 0
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELASTICITY_CLASS = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FLUID_MECHANICS_CLASS = 2
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELECTROMAGNETICS_CLASS = 3
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_CLASSICAL_FIELD_CLASS = 4
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MODAL_CLASS = 5
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FITTING_CLASS = 6
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_OPTIMISATION_CLASS = 7
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_TYPE = 0
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_ELASTICITY_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SETFINITE_ELASTICITY_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STOKES_FLUID_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NAVIER_STOKES_FLUID_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELECTROSTATIC_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MAGNETOSTATIC_TYPE = 2
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MAXWELLS_EQUATIONS_TYPE = 3
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LAPLACE_EQUATION_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_POISSON_EQUATION_TYPE = 2

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
HELMHOLTZ_EQUATION_TYPE = 3
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_WAVE_-
EQUATION_TYPE = 4
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
DIFFUSION_EQUATION_TYPE = 5
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
ADVECTION_DIFFUSION_EQUATION_TYPE = 6
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
REACTION_DIFFUSION_EQUATION_TYPE = 7
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
BIHARMONIC_EQUATION_TYPE = 8
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR_-
ELASTIC_MODAL_TYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_-
SUBTYPE = 0
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
STANDARD_LAPLACE_SUBTYPE = 1
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_-
GENERALISED_LAPLACE_SUBTYPE = 2
- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
INITIAL_TYPE = 1

Initial setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
GEOMETRY_TYPE = 2

Geometry setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
DEPENDENT_TYPE = 3

Dependent variables.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
MATERIALS_TYPE = 4

Materials setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
SOURCE_TYPE = 5

Source setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
SOURCE_MATERIALS_TYPE = 6

Source materials setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
ANALYTIC_TYPE = 7

Analytic setup.

- INTEGER(INTG), parameter EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_-
FIXED_CONDITIONS_TYPE = 8

Fixed conditions.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_EQUATIONS_TYPE = 9**
Equations setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINAL_TYPE = 9**
Final setup.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_START_ACTION = 1**
Start setup action.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SETUP_FINISH_ACTION = 2**
Finish setup action.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NOT_FIXED = 0**
The dof is not fixed.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FIXED_BOUNDARY_CONDITION = 1**
The dof is fixed as a boundary condition.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MIXED_BOUNDARY_CONDITION = 2**
The dof is set as a mixed boundary condition.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_LINEARITY_TYPES = 3**
The number of problem linearity types defined.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_LINEAR = 1**
The problem is linear.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR = 2**
The problem is non-linear.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NONLINEAR_BCS = 3**
The problem has non-linear boundary conditions.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_TIME_TYPES = 3**
The number of problem time dependence types defined.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_STATIC = 1**

The problem is static and has no time dependence.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_DYNAMIC = 2**

The problem is dynamic.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_QUASISTATIC = 3**

The problem is quasi-static.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::NUMBER_OF_EQUATIONS_SET_SOLUTION_METHODS = 6**

The number of solution methods defined.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FEM_SOLUTION_METHOD = 1**

Finite Element Method solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_BEM_SOLUTION_METHOD = 2**

Boundary Element Method solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FD_SOLUTION_METHOD = 3**

Finite Difference solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FV_SOLUTION_METHOD = 4**

Finite Volume solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFEM_SOLUTION_METHOD = 5**

Grid-based Finite Element Method solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_GFV_SOLUTION_METHOD = 6**

Grid-based Finite Volume solution method.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_NO_OUTPUT = 0**

No output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_TIMING_OUTPUT = 1**

Timing information output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_MATRIX_OUTPUT = 2**

All below and equation matrices output.

- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_ELEMENT_MATRIX_OUTPUT = 3**
All below and Element matrices output.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_SPARSE_MATRICES = 1**
Use sparse matrices for the equations set.
- INTEGER(INTG), parameter **EQUATIONS_SET_CONSTANTS::EQUATIONS_SET_FULL_MATRICES = 2**
Use fully populated matrices for the equations set.

8.23.1 Detailed Description

Id

[equations_set_constants.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all constants shared across equations set routines.

8.23.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [equations_set_constants.f90](#).

8.24 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/equations_set_routines.f90 File Reference

Id

[equations_set_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- interface [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_OF](#)
- interface [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICATION_SET](#)

Namepaces

- namespace [EQUATIONS_SET_ROUTINES](#)

This module handles all equations set routines.

Functions

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_FINISH](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Finish the creation of a analytic solution for equations set.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_CREATE_START](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Start the creation of a analytic solution for a equations set.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_DESTROY](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Destroy the analytic solution for an equations set.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_FINALISE](#) ([EQUATIONS_SET_ANALYTIC](#), ERR, ERROR,*)

Finalise the analytic solution for an equations set and deallocate all memory.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ANALYTIC_INITIALISE](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Initialises the analytic solution for an equations set.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Assembles the equations for an equations set.

- subroutine [EQUATIONS_SET_ROUTINES::EQUATIONS_SET_ASSEMBLE_LINEAR_STATIC_FEM](#) ([EQUATIONS_SET](#), ERR, ERROR,*)

Assembles the equations stiffness matrix and rhs for a linear static equations set using the finite element method.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_BACKSUBSTITUTE** (EQUATIONS_SET, ERR, ERROR,*)

Backsubstitutes with an equations set to calculate unknown right hand side vectors.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finishes the process of creating an equation set on a region.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_CREATE_START** (USER_NUMBER, REGION, GEOM_FIBRE_FIELD, EQUATIONS_SET, ERR, ERROR,*)

Starts the process of creating an equations set defined by USER_NUMBER in the region identified by REGION.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DESTROY** (USER_NUMBER, REGION, ERR, ERROR,*)

Destroys an equations set identified by a user number on the give region and deallocates all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FINALISE** (EQUATIONS_SET, ERR, ERROR,*)

Finalise the equations set and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SETFINITE_ELEMENT_Calculate** (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matries and rhs vector for the given element number for a finite element equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_FINALISE** (EQUATIONS_INTERPOLATION, ERR, ERROR,*)

Finalises the interpolation information for equations and deallocates all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_INTERPOLATION_INITIALISE** (EQUATIONS, ERR, ERROR,*)

Initialises the interpolation information for equations.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_APPLY** (EQUATIONS_SET, ERR, ERROR,*)

Applies the fixed conditions in an equation set to the dependent field in an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of fixed conditions for an equation set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of fixed conditions for a problem.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)
Destroy the fixed conditions for an equations set and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_FINALISE** (EQUATIONS_SET_FIXED_CONDITIONS, ERR, ERROR,*)
Finalise the fixed conditions for an equations set and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)
Initialises the fixed conditions for a problem.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOFS** (EQUATIONS_SET, DOF_INDICES, CONDITIONS, VALUES, ERR, ERROR,*)
Sets fixed conditions for the equations set on the specified dofs.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_FIXED_CONDITIONS_SET_DOF1** (EQUATIONS_SET, DOF_INDEX, CONDITION, VALUE, ERR, ERROR,*)
Sets a fixed condition for the equation set on the specified dof.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_GEOMETRY_FINALISE** (EQUATIONS_SET_GEOMETRY, ERR, ERROR,*)
Finalise the geometry for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_GEOMETRY_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)
Initialises the geometry for an equation set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_LINEAR_DATA_FINALISE** (EQUATIONS_LINEAR_DATA, ERR, ERROR,*)
Finalise the equations set linear data and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUIVARIANT_LINEAR_DATA_INITIALISE** (EQUATIONS, ERR, ERROR,*)
Initialises the linear data information for an equations.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_INTERPOLATION_SET** (EQUATIONS_SET, COMPONENT_NUMBER, INTERPOLATION_TYPE, ERR, ERROR,*)
Sets/changes the field component interpolation for a materials field of a problem.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_COMPONENT_MESH_COMPONENT_SET** (EQUATIONS_SET, COMPONENT_NUMBER, MESH_COMPONENT_NUMBER, ERR, ERROR,*)
Sets/changes the field component mesh component for a materials field of a problem.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)
Finish the creation of materials for an equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of materials for a problem.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)

Destroy the materials for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_FINALISE** (EQUATIONS_SET_MATERIALS, ERR, ERROR,*)

Finalise the materials for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the materials for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_MATERIAL_FIELD_GET** (EQUATIONS_SET, MATERIAL_FIELD, ERR, ERROR,*)

Returns a pointer to the material field of the materials for a problem.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_MATERIALS_SCALING_SET** (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a materials field of an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_FINALISE** (EQUATIONS_NONLINEAR_DATA, ERR, ERROR,*)

Finalise the equations set nonlinear data and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_NONLINEAR_DATA_INITIALISE** (EQUATIONS, ERR, ERROR,*)

Initialises the nonlinear data information for an equations.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_** (EQUATIONS_SET, VARIABLE_NUMBER, COMPONENT_NUMBER,&MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Sets/changes the field component mesh component for a dependent field of an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of a dependent variables for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of dependent variables for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_DEPENDENT_FIELD_GET** (EQUATIONS_SET, DEPENDENT_FIELD, ERR, ERROR,*)

Returns a pointer to the dependent field of the dependent variables for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)

Destroy the dependent variables for an equations sety.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_FINALISE** (EQUATIONS_SET_DEPENDENT, ERR, ERROR,*)

Finalises the dependent variables for an equation set and deallocates all memory.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the dependent variables for a equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_DEPENDENT_SCALING_SET** (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a dependent field of an equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SETUP** (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the specifics for an equation set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of equations for the equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of equations for the equation set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_FINALISE** (EQUATIONS, ERR, ERROR,*)

Finalise the equations and deallocate all memory.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_SPARSITY_TYPE_SET** (EQUATIONS_SET, SPARSITY_TYPE, ERR, ERROR,*)

Sets/changes the sparsity type for the equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the equations for an equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_OUTPUT_TYPE_SET** (EQUATIONS_SET, OUTPUT_TYPE, ERR, ERROR,*)

Sets/changes the output type for the equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_FINISH** (EQUATIONS_SET, ERR, ERROR,*)

Finish the creation of a source for an equation set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_CREATE_START** (EQUATIONS_SET, ERR, ERROR,*)

Start the creation of a source for an equations set.

- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_DESTROY** (EQUATIONS_SET, ERR, ERROR,*)

Destroy the source for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_FINALISE** (EQUATIONS_SET_SOURCE, ERR, ERROR,*)

Finalise the source for a equations set and deallocate all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_INITIALISE** (EQUATIONS_SET, ERR, ERROR,*)

Initialises the source for an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SOURCE_SCALING_SET** (EQUATIONS_SET, SCALING_TYPE, ERR, ERROR,*)

Sets/changes the field scaling for a source field of an equations set.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_SPECIFICAT** (USER_NUMBER, REGION, EQUATIONS_SET_CLASS, EQUATIONS_SET_TYPE_, &EQUATIONS_SET_SUBTYPE, ERR, ERROR,*)

Sets/changes the equation set specification i.e., equation set class, type and subtype for an equation set identified by a user number.
- subroutine **EQUATIONS_SET_ROUTINES::EQ** (EQUATIONS_SET, EQUATIONS_SET_CLASS, EQUATIONS_SET_TYPE_, EQUATIONS_SET_SUBTYPE, &ERR, ERROR,*)

Sets/changes the equations set specification i.e., equations set class, type and subtype for a equations set identified by a pointer.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_DATA_FINALISE** (TIME_DATA, ERR, ERROR,*)

Finalises the time data information for an equations and deallocates all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_EQUATIONS_TIME_DATA_INITIALISE** (EQUATIONS, ERR, ERROR,*)

Initialises the time data information for an equations.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SET_USER_NUMBER_FIND** (USER_NUMBER, REGION, EQUATIONS_SET, ERR, ERROR,*)

Finds and returns in EQUATIONS_SET a pointer to the equations set identified by USER_NUMBER in the given REGION. If no equations set with that USER_NUMBER exists EQUATIONS_SET is left nullified.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_FINALISE** (REGION, ERR, ERROR,*)

Finalises all equations sets on a region and deallocates all memory.
- subroutine **EQUATIONS_SET_ROUTINES::EQUATIONS_SETS_INITIALISE** (REGION, ERR, ERROR,*)

Initialises all equations sets on a region.

8.24.1 Detailed Description

Id

[equations_set_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all equations set routines.

8.24.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [equations_set_routines.f90](#).

8.25 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_c.c File Reference

Include dependency graph for f90c_c.c:

Typedefs

- `typedef int integer`
- `typedef integer logical`

Functions

- `void CStringLen (int *length, char *string)`
- `void PackCharacters (integer *integer_char, integer *char_num, char *integer_string)`
- `void UnPackCharacters (integer *integer_char, integer *char_num, char *integer_string)`

8.25.1 Typedef Documentation

8.25.1.1 `typedef int integer`

Definition at line 98 of file f90c_c.c.

8.25.1.2 `typedef integer logical`

Definition at line 99 of file f90c_c.c.

8.25.2 Function Documentation

8.25.2.1 `void CStringLen (int * length, char * string)`

Definition at line 118 of file f90c_c.c.

8.25.2.2 `void PackCharacters (integer * integer_char, integer * char_num, char * integer_string)`

Definition at line 124 of file f90c_c.c.

8.25.2.3 `void UnPackCharacters (integer * integer_char, integer * char_num, char * integer_string)`

Definition at line 140 of file f90c_c.c.

8.26 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/f90c_f.f90 File Reference

Id

[f90c_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [F90C::interface](#)

Namespaces

- namespace [F90C](#)

This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c_c.c](#) module.

Functions

- INTEGER(INTG) [F90C::CSTRINGLENGTH](#) (CSTRING)
- subroutine [F90C::C2FSTRING](#) (CSTRING, FSTRING, ERR, ERROR,*)
- INTEGER(INTG) [F90C::FSTRINGLENGTH](#) (FSTRING)
- subroutine [F90C::F2CSTRING](#) (CSTRING, FSTRING, ERR, ERROR,*, FSTRINGLEN)

8.26.1 Detailed Description

Id

[f90c_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module handles calling C from Fortran90 and vise versa. It needs to be linked with the [f90c_c.c](#) module.

8.26.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [f90c_f.f90](#).

8.27 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_IO_routines.f90 File Reference

Id

parallel_IO.f90 1 2009-02-19 13:57:57Z cpb

Classes

- struct [FIELD_IO_ROUTINES::MESH_ELEMENTS_TYPE_PTR_TYPE](#)
field variable compoment type pointer for IO
- struct [FIELD_IO_ROUTINES::FIELD_VARIABLE_COMPONENT_PTR_TYPE](#)
field variable compoment type pointer for IO
- struct [FIELD_IO_ROUTINES::FIELD_IO_INFO_SET](#)
contains information for parallel IO, and it is nodal base
- struct [FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET](#)
contains information for parallel IO, and it is nodal base
- struct [FIELD_IO_ROUTINES::FIELD_IO_ELEMENTALL_INFO_SET](#)
contains information for parallel IO, and it is nodal base

Namespaces

- namespace [FIELD_IO_ROUTINES](#)
Implements lists of Field IO operation.

Functions

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_FIELD_INFO](#) (STRING, LABEL_TYPE, FIELD_TYPE, ERR, ERROR,*)
Get the field information.
- INTEGER(INTG) [FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_INFO](#) (LINE, ERR, ER-ROR)
Get the derivative information.
- subroutine [FIELD_IO_ROUTINES::FIELD_IO_CREATE_FIELDS](#)
Create decompositon.
- subroutine [FIELD_IO_ROUTINES::FIE](#) (DECOMPOSITION, DECOMPOSITION_USER_-NUMBER, DECOMPOSITION_METHOD, MESH, NUMBER_OF_DOMAINS,&ERR, ERROR,*)
Create decompiton.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_FILEDS_IMPORT](#) (NAME, METHOD, REGION, MESH, MESH_USER_NUMBER, DECOMPOSITION, DECOMPOSITION_USER_NUMBER,&DECOMPOSITION_METHOD, FIELD_VALUES_SET_TYPE, FIELD_SCALING_TYPE, ERR, ERROR,*)

Import fields from files into different computational nodes.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_FILL_BASIS_INFO](#) (INTERPOLATION_XI, LIST_STR, NUMBER_OF_COMPONENTS, ERR, ERROR,*)

Finding basis information.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_IMPORT_GLOBAL_MESH](#) (NAME, REGION, MESH, MESH_USER_NUMBER, MA TER_COMPUTATIONAL_NUMBER,&my_computational_node_number,&MESH_COMPONENTS_OF_FIELD_COMPONENTS,&COMPONENTS_IN_FIELDS, NUMBER_OF_FIELDS, NUMBER_OF_EXNODE_FILES, ERR, ERROR,*)

Read the global mesh into one computational node first and then broadcasting to others nodes.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_TRANSLATE_LABEL_INTO_INTERPOLATION_TYPE](#) (INTERPOLATION, LABEL_TYPE, ERR, ERROR,*)

Finding basis information.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_ELEMENTS_EXPORT](#) (FIELDS, FILE_NAME, METHOD, ERR, ERROR,*)

Export elemental information into multiple files.

- TYPE(VARYING_STRING) [FIELD_IO_ROUTINES::FIELD_IO_BASIS_LHTP_FAMILY_LABEL](#) (BASIS, num_scl, num_node, LABEL_TYPE, ERR, ERROR)

Finding basis information.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTAL_GROUP_HEADER_FORTRAN](#) (LOCAL_PROCESS_ELEMENTAL_INFO_SET, LOCAL ELEMENTAL_NUMBER, MAX_NODE_CPMP_INDEX,&NUM_OF_SCALING_FACTOR_SETS, LIST_COMP_SCALE, my_computational_node_number, FILE_ID, ERR, ERROR,*)

Write the header of a group elements using FORTRAN.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_EXPORT_ELEMENTS_INTO_](#) (LOCAL_PROCESS_ELEMENTAL_INFO_SET, NAME, my_computational_node_number,&computational_node_numbers, ERR, ERROR,*)

Write all the elemental information from LOCAL_PROCESS_NODAL_INFO_SET to exelem files.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_SORT](#) (LOCAL_PROCESS_ELEMENTAL_INFO_SET, my_computational_node_number, ERR, ERROR,*)

Sort the Elemental_info_set according to the type of field variable components.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_ATTACH_LOCAL_PROCESS](#) (LOCAL_PROCESS_ELEMENTAL_INFO_SET, ERR, ERROR,*)

Collect the elemental information from each MPI process.

- subroutine [FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_FINALIZE](#) (LOCAL_PROCESS_ELEMENTAL_INFO_SET, ERR, ERROR,*)

Finalized the elemental information set.

- subroutine **FIELD_IO_ROUTINES::FIELD_IO_ELEMENTAL_INFO_SET_INITIALISE** (LOCAL_PROCESS_ELEMENTAL_INFO_SET, FIELDS, ERR, ERROR,*)
Initialize the elemental information set.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_NODES_EXPORT** (FIELDS, FILE_NAME, METHOD, ERR, ERROR,*)
Export nodal information.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_INITIALISE** (LOCAL_PROCESS_NODAL_INFO_SET, FIELDS, ERR, ERROR,*)
Initialize nodal information set.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS** (LOCAL_PROCESS_NODAL_INFO_SET, ERR, ERROR,*)
Collect nodal information from each MPI process.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_SORT** (LOCAL_PROCESS_NODAL_INFO_SET, my_computational_node_number, ERR, ERROR,*)
Sort nodal information according to the type of field variable component.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_NODAL_INFO_SET_FINALIZE** (LOCAL_PROCESS_NODAL_INFO_SET, ERR, ERROR,*)
Finalize nodal information set.
- TYPE(VARYING_STRING) **FIELD_IO_ROUTINES::FIELD_IO_LABEL_DERIVATIVE_INFO_GET** (GROUP_DERIVATIVES, NUMBER_DERIVATIVES, LABEL_TYPE, ERR, ERROR)
Get the derivative information.
- TYPE(VARYING_STRING) **FIELD_IO_ROUTINES::FIELD_IO_LABEL_FIELD_INFO_GET** (COMPONENT, LABEL_TYPE, ERR, ERROR)
Get the field information.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODAL_GROUP_HEADER_FORTRAN** (LOCAL_PROCESS_NODAL_INFO_SET, LOCAL_NODAL_NUMBER, MAX_NUM_OF_NODAL_DERIVATIVES, &my_computational_node_number, FILE_ID, ERR, ERROR,*)
Write the header of a group nodes using FORTRAN.
- subroutine **FIELD_IO_ROUTINES::FIELD_IO_EXPORT_NODES_INTO_LOC** (LOCAL_PROCESS_NODAL_INFO_SET, NAME, my_computational_node_number, &computational_node_numbers, ERR, ERROR,*)
Write all the nodal information from LOCAL_PROCESS_NODAL_INFO_SET to local exnode files.
- TYPE(VARYING_STRING) **FIELD_IO_ROUTINES::FIELD_IO_FILEDS_GROUP_INFO_GET** (FIELDS, ERR, ERROR)
Get the region label.
- TYPE(VARYING_STRING) **FIELD_IO_ROUTINES::FIELD_IO_MULTI_FILES_INFO_GET** (computational_node_numbers, ERR, ERROR)

Get the number of files.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_STRING` (FILE_ID, STRING_DATA, FILE_END, ERR, ERROR,*)

Read a string using FORTRAN IO.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_STRING` (FILE_ID, STRING_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a string using FORTRAN IO.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_DP` (FILE_ID, REAL_DATA, LEN_OF_DATA, FILE_END, ERR, ERROR,*)

Read a real data using FORTRAN IO.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_DP` (FILE_ID, REAL_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a real data using FORTRAN IO.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_READ_INTG` (FILE_ID, INTG_DATA, LEN_OF_DATA, ERR, ERROR,*)

Read a integer data.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_WRITE_INTG` (FILE_ID, INTG_DATA, LEN_OF_DATA, ERR, ERROR,*)

Write a integer data.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_OPEN` (FILE_ID, FILE_NAME, FILE_STATUS, ERR, ERROR,*)

Open a file using Fortran.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_REWIND` (FILE_ID, ERR, ERROR,*)

Close a file using FORTRAN IO to rewind.

- subroutine `FIELD_IO_ROUTINES::FIELD_IO_FORTRAN_FILE_CLOSE` (FILE_ID, ERR, ERROR,*)

Close a file using Fortran.

- subroutine `FIELD_IO_ROUTINES::STRING_TO_MUTI_INTEGERS_VS` (STRING, NUMBER_OF_INTEGERS, INTG_DATA, ERR, ERROR,*)

- subroutine `FIELD_IO_ROUTINES::STRING_TO_MUTI_REALS_VS` (STRING, NUMBER_OF_REALS, REAL_DATA, POSITION, ERR, ERROR,*)

Variables

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::SHAPE_SIZE` = 3

size of shape

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_FIELD_LABEL` = 1

Type for label.

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_VARIABLE_LABEL` = 2
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_COMPONENT_LABEL` = 3
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_DERIVATIVE_LABEL` = 4
- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_SCALE_FACTORS_NUMBER_TYPE` = 5

Type of scale factor.

- INTEGER(INTG), parameter `FIELD_IO_ROUTINES::FIELD_IO_SCALE_FACTORS_PROPERTY_TYPE` = 6

8.27.1 Detailed Description

Id

parallel_IO.f90 1 2009-02-19 13:57:57Z cpb

Author:

Heye Zhang ThiS module handles parallel Io. Using mpi2 and parall print function, formatted text and binary IO are supported in openCMISS

8.27.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [field_IO_routines.f90](#).

8.28 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/field_routines.f90 File Reference

Id

[field_routines.f90](#) 29 2007-08-09 16:19:43Z cpb

Classes

- interface [FIELD_ROUTINES::FIELD_COMPONENT_INTERPOLATION_SET](#)
- interface [FIELD_ROUTINES::FIELD_COMPONENT_MESH_COMPONENT_SET](#)
- interface [FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET](#)
- interface [FIELD_ROUTINES::FIELD_DIMENSION_SET](#)
- interface [FIELD_ROUTINES::FIELD_GEOMETRIC_FIELD_SET](#)
- interface [FIELD_ROUTINES::FIELD_MESH_DECOMPOSITION_SET](#)
- interface [FIELD_ROUTINES::FIELD_NUMBER_OF_COMPONENTS_SET](#)
- interface [FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLES_SET](#)
- interface [FIELD_ROUTINES::FIELD_SCALING_TYPE_SET](#)
- interface [FIELD_ROUTINES::FIELD_TYPE_SET](#)

Namespaces

- namespace [FIELD_ROUTINES](#)

This module handles all field related routines.

Functions

- subroutine [FIELD_ROUTINES::FIELD_COMPONENT_INTER](#) (USER_NUMBER, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, REGION,&INTERPOLATION_TYPE, ERR, ERROR,*)

Sets/changes the interpolation type for a field variable component identified by a user number and component number on a region.
- subroutine [FIELD_ROUTINES::FI](#) (FIELD, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, INTERPOLATION_TYPE,&ERR, ERROR,*)

Sets/changes the interpolation type for a field variable component identified by a pointer.
- subroutine [FIELD_ROUTINES::FIELD_COMPONENT_MESH_COM](#) (USER_NUMBER, FIELD_VARIABLE_NUMBER, FIELD_COMPONENT_NUMBER, REGION,&MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Sets/changes the mesh component number for a field variable component identified by a user number, component number and variable number on a region.
- subroutine [FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_FINALISE](#) (FIELD_VARIABLE_COMPONENT, ERR, ERROR,*)

Finalises a field variable component and deallocates all memory.
- subroutine [FIELD_ROUTINES::FIELD_VARIABLE_COMPONENT_INITIALISE](#) (FIELD, VARIABLE_NUMBER, COMPONENT_NUMBER, ERR, ERROR,*)

Initialises a field variable component.

- subroutine **FIELD_ROUTINES::FIELD_CREATE_FINISH** (REGION, FIELD, ERR, ERROR,*)

Finishes the creation of a field on a region.
- subroutine **FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_FINALISE** (FIELD, ERR, ERROR,*)

Finalise the create values cache for a field.
- subroutine **FIELD_ROUTINES::FIELD_CREATE_VALUES_CACHE_INITIALISE** (FIELD, ERR, ERROR,*)

Initialises the create values cache for a field.
- subroutine **FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_NUMBER** (USER_NUMBER, REGION, DEPENDENT_TYPE, ERR, ERROR,*)

Sets/changes the dependent type for a field identified by a user number.
- subroutine **FIELD_ROUTINES::FIELD_DEPENDENT_TYPE_SET_PTR** (FIELD, DEPENDENT_TYPE, ERR, ERROR,*)

Sets/changes the dependent type for a field indentified by a pointer.
- subroutine **FIELD_ROUTINES::FIELD_DESTROY** (FIELD, ERR, ERROR,*)

Destroys a field identified by a pointer to a field.
- subroutine **FIELD_ROUTINES::FIELD_DIMENSION_SET_NUMBER** (USER_NUMBER, REGION, FIELD_DIMENSION, ERR, ERROR,*)

Sets/changes the field dimension for a field identified by a user number.
- subroutine **FIELD_ROUTINES::FIELD_DIMENSION_SET_PTR** (FIELD, FIELD_DIMENSION, ERR, ERROR,*)

Sets/changes the field dimension for a field identified by a pointer.
- subroutine **FIELD_ROUTINES::FIELD_INTERPOLATE_GAUSS** (PARTIAL_DERIVATIVE_TYPE, QUADRATURE_SCHEME, GAUSS_POINT_NUMBER, INTERPOLATED_POINT, ERR, ERROR,*)

Interpolates a field at a gauss point to give an interpolated point. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name XEXG, ZEXG.
- subroutine **FIELD_ROUTINES::FIELD_INTERPOLATE_XI** (PARTIAL_DERIVATIVE_TYPE, XI, INTERPOLATED_POINT, ERR, ERROR,*)

Interpolates a field at a xi location to give an interpolated point. XI is the element location to be interpolated at. PARTIAL_DERIVATIVE_TYPE controls which partial derivatives are evaluated. If it is NO_PART_DERIV then only the field values are interpolated. If it is FIRST_PART_DERIV then the field values and first partial derivatives are interpolated. If it is SECOND_PART_DERIV the the field values and first and second partial derivatives are evaluated. Old CMISS name PXI.
- subroutine **FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_FINALISE** (INTERPOLATED_POINT, ERR, ERROR,*)

Finalises the interpolated point and deallocates all memory.

- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_INITIALISE`**
(INTERPOLATION_PARAMETERS, INTERPOLATED_POINT, ERR, ERROR,*)

Initialises the interpolated point for an interpolation parameters.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_CALCULATE`**
(JACOBIAN_TYPE, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Calculates the interpolated point metrics and the associated interpolated point.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_FINALISE`**
(INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Finalises the interpolated point metrics and deallocates all memory.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATED_POINT_METRICS_INITIALISE`**
(INTERPOLATED_POINT, INTERPOLATED_POINT_METRICS, ERR, ERROR,*)

Initialises the interpolated point metrics for an interpolated point.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_ELEMENT_GET`**
(PARAMETER_SET_NUMBER, ELEMENT_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Gets the interpolation parameters for a particular element. Old [CMISS](#) name XPXE, ZPZE.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_FINALISE`**
(INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Finalises the interpolation parameters and deallocates all memory.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_INITIALISE`**
(FIELD, VARIABLE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Initialises the interpolation parameters for a field variable.
- subroutine **`FIELD_ROUTINES::FIELD_INTERPOLATION_PARAMETERS_LINE_GET`**
(PARAMETER_SET_NUMBER, LINE_NUMBER, INTERPOLATION_PARAMETERS, ERR, ERROR,*)

Gets the interpolation parameters for a particular line. Old [CMISS](#) name XPXE, ZPZE.
- subroutine **`FIELD_ROUTINES::FIELD_VARIABLES_FINALISE`**(FIELD, ERR, ERROR,*)

Finalises the field variables for a field and deallocates all memory.
- subroutine **`FIELD_ROUTINES::FIELD_VARIABLES_INITIALISE`**(FIELD, ERR, ERROR,*)

Initialises the field variables.
- subroutine **`FIELD_ROUTINES::FIELDS_FINALISE`**(REGION, ERR, ERROR,*)

Finalises the fields for the given region and deallocates all memory.
- subroutine **`FIELD_ROUTINES::FIELDS_INITIALISE`**(REGION, ERR, ERROR,*)

Initialises the fields for the given region.

Variables

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_INDEPENDENT_TYPE** = 1
Independent field type.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_DEPENDENT_TYPE** = 2
Dependent field type.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_SCALAR_DIMENSION_TYPE** = 1
Scalar field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_VECTOR_DIMENSION_TYPE** = 2
Vector field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_GEOMETRIC_TYPE** = 1
Geometric field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_FIBRE_TYPE** = 2
Fibre field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_GENERAL_TYPE** = 3
General field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_MATERIAL_TYPE** = 4
Material field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_CONSTANT_INTERPOLATION** = 1
Constant interpolation. One parameter for the field.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_ELEMENT_BASED_INTERPOLATION** = 2
Element based interpolation. Parameters are different in each element.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NODE_BASED_INTERPOLATION** = 3
Node based interpolation. Parameters are nodal based and a basis function is used.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_POINT_BASED_INTERPOLATION** = 4
Point based interpolation. Parameters are different at each grid point.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NUMBER_OF_VARIABLE_TYPES** = 4
Number of different field variable types possible.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_STANDARD_VARIABLE_TYPE** = 1
Standard variable type i.e., u.
- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NORMAL_VARIABLE_TYPE** = 2

Normal derivative variable type i.e., du/dn .

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_TIME_DERIV1_VARIABLE_TYPE** = 3

First time derivative variable type i.e., du/dt .

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_TIME_DERIV2_VARIABLE_TYPE** = 4

Second type derivative variable type i.e., d^2u/dt^2 .

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_CONSTANT_DOF_TYPE** = 1

The dof is from a field variable component with constant interpolation.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_ELEMENT_DOF_TYPE** = 2

The dof is from a field variable component with element based interpolation.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NODE_DOF_TYPE** = 3

The dof is from a field variable component with node based interpolation.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_POINT_DOF_TYPE** = 4

The dof is from a field variable component with point based interpolation.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NUMBER_OF_SET_TYPES** = 99

The maximum number of different parameter sets for a field.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_VALUES_SET_TYPE** = 1

The parameter set corresponding to the field values.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_BOUNDARY_CONDITIONS_SET_TYPE** = 2

The parameter set corresponding to the field boundary conditions.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_INITIAL_CONDITIONS_SET_TYPE** = 3

The parameter set corresponding to the field initial conditions.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_NO_SCALING** = 0

The field is not scaled.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_UNIT_SCALING** = 1

The field has unit scaling.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_ARC_LENGTH_SCALING** = 2

The field has arc length scaling.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_ARITHMETIC_MEAN_SCALING** = 3

The field has arithmetic mean of the arc length scaling.

- INTEGER(INTG), parameter **FIELD_ROUTINES::FIELD_HARMONIC_MEAN_SCALING** = 4

The field has geometric mean of the arc length scaling.

8.28.1 Detailed Description

Id

[field_routines.f90](#) 29 2007-08-09 16:19:43Z cpb

Author:

Chris Bradley This module handles all field related routines.

8.28.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [field_routines.f90](#).

8.29 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite_element_routines.f90 File Reference

Id

[finite_element_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [FINITE_ELEMENT_ROUTINES](#)
This module contains all finite element base routines.

Functions

- subroutine [FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_FINALISE](#)(ELEMENT_MATRICES, ERR, ERROR,*)
- subroutine [FINITE_ELEMENT_ROUTINES::FEM_ELEMENT_MATRICES_INITIALISE](#)(PROBLEM, ELEMENT_MATRICES, ERR, ERROR,*)

8.29.1 Detailed Description

Id

[finite_element_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all finite element base routines.

8.29.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and

8.29 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/finite_element_routines.f90 File Reference

not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [finite_element_routines.f90](#).

8.30 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/fluid_- mechanics_routines.f90 File Reference

Id

[fluid_mechanics_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [FLUID_MECHANICS_ROUTINES](#)

This module handles all fluid mechanics class routines.

8.30.1 Detailed Description

Id

[fluid_mechanics_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all fluid mechanics routines.

8.30.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [fluid_mechanics_routines.f90](#).

8.31 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/generated_mesh_routines.f90 File Reference

Id

[generated_mesh_routines.f90](#) 9 2007-05-15 13:52:02Z cpb

Namespaces

- namespace [GENERATED_MESH_ROUTINES](#)

This module handles all generated mesh routines.

Functions

- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_FINISH](#)([GENERATED_MESH](#), [ERR](#), [ERROR,*](#))
Finishes the creation of a generated mesh.
- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_CREATE_START](#)([USER_NUMBER](#), [REGION](#), [GENERATED_MESH](#), [ERR](#), [ERROR,*](#))
Starts the creation of a generated mesh.
- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_DESTROY](#)([GENERATED_MESH](#), [ERR](#), [ERROR,*](#))
Destroys a generated mesh.
- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_FINALISE](#)([GENERATED_MESH](#), [ERR](#), [ERROR,*](#))
Finalises a generated mesh and deallocates all memory.
- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_INITIALISE](#)([GENERATED_MESH](#), [ERR](#), [ERROR,*](#))
Initialises a generated mesh.
- subroutine [GENERATED_MESH_ROUTINES::GENERATED_MESH_TYPE_SET](#)([GENERATED_MESH](#), [GENERATED_TYPE](#), [ERR](#), [ERROR,*](#))
Sets/changes the type of a generated mesh.

Variables

- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_REGULAR_MESH_TYPE](#) = 1
A regular generated mesh.
- INTEGER(INTG), parameter [GENERATED_MESH_ROUTINES::GENERATED_MESH_POLAR_MESH_TYPE](#) = 2
A polar generated mesh.

- INTEGER(INTG), parameter GENERATED_MESH_ROUTINES::GENERATED_MESH_-
FRACTAL_TREE_MESH_TYPE = 3

A fractal tree generated mesh.

8.31.1 Detailed Description

Id

[generated_mesh_routines.f90](#) 9 2007-05-15 13:52:02Z cpb

Author:

Chris Bradley This module handles all generated mesh routines.

Todo

Move generated regular mesh from mesh routines and generalise.

8.31.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [generated_mesh_routines.f90](#).

8.32 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/input_output.f90 File Reference

Id

[input_output.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [INPUT_OUTPUT::WRITE_STRING](#)
Write a string to a given output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_VALUE](#)
Write a string followed by a value to a given output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE](#)
Write a string, value, string then a value to a given output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_FMT_VALUE](#)
Write a string followed by a value formatted in a particular way to a specified output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_FMT_TWO_VALUE](#)
Write a string, value, string then a value with the values formatted in a particular way to a given output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_VECTOR](#)
Write a string followed by a vector to a specified output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_IDX_VECTOR](#)
Write a string followed by a indexed vector to a specified output stream.
- interface [INPUT_OUTPUT::WRITE_STRING_MATRIX](#)
Write a string followed by a matrix to a specified output stream.

Namespaces

- namespace [INPUT_OUTPUT](#)
This module handles all formating and input and output.

Functions

- subroutine [INPUT_OUTPUT::WRITE_STRING_C](#) (ID, STRING, ERR, ERROR,*)
Writes the character STRING to the given output stream specified by ID.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VS](#) (ID, STRING, ERR, ERROR,*)
Writes the varying string STRING to the given output stream specified by ID.

- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_C](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_DP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_INTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_LINTG](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_L](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_SP](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_VALUE_VS](#) (ID, FIRST_STRING, VALUE, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. Free format is used to format the value.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_C_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_DP_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_L](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_SP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_INTG_VS](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_C](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_DP](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_INTG](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_SP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_L_VS` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_C` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_DP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_INTG` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_L` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_SP` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine `INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_SP_VS` (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.

- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_C**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_DP**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_INTG**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_L**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_SP**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_TWO_VALUE_VS_VS**](#) (ID, FIRST_STRING, FIRST_VALUE, SECOND_STRING, SECOND_VALUE, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. Free format is used to format both values.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_C**](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_DP**](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine [**INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_INTG**](#) (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.

- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_LINTG` (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_L` (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_SP` (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine `INPUT_OUTPUT::WRITE_STRING_FMT_VALUE_VS` (ID, FIRST_STRING, VALUE, FORMAT_STRING, ERR, ERROR,*)

Writes the FIRST STRING followed by a formatted character VALUE to the given output stream specified by ID. FORMAT_STRING is used to format the value.
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formmatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formmatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formmatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.
- subroutine `INPUT_OUTPUT::WR` (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formmatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted character FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WRITE_STRING_FMT**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted double precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [**INPUT_OUTPUT::WR**](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WRITE_STRING_FMT](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted integer FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted logical FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formmatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formmatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formmatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted single precision FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted character SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted double precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WRITE_STRING_FMT** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE,&SECOND_FORMAT, ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted integer SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted logical SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted single precision SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine **INPUT_OUTPUT::WR** (ID, FIRST_STRING, FIRST_VALUE, FIRST_FORMAT, SECOND_STRING, SECOND_VALUE, SECOND_FORMAT,&ERR, ERROR,*)

Writes the FIRST_STRING followed by a formatted varying string FIRST_VALUE and the the SECOND_STRING followed by a formatted varying string SECOND_VALUE to the given output stream specified by ID. FIRST_FORMAT is used to format the first value and SECOND_FORMAT is used to format the second value.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given double precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given integer VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given logical VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [INPUT_OUTPUT::WR](#) (ID, FIRST_IDX, DELTA, LAST_IDX, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT, REPEAT_FORMAT,&ERR, ERROR,*)

Writes the given single precision VECTOR to the given output stream specified by ID. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_IDX and LAST_IDX are the extents of the data and DELTA is the NUMBER of indices to skip for each index.

- subroutine [INPUT_OUTPUT::WRITE_STRING_IDX](#) (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed double precision VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_IDX` (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_IDX` (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed integer VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_IDX` (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed logical VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_IDX` (ID, NUM_INDICES, INDICES, DELTA, NUMBER_FIRST, NUMBER_REPEAT, VECTOR, FIRST_FORMAT,&REPEAT_FORMAT, ERR, ERROR,*)

Writes the given indexed single precision VECTOR to the given output stream specified by ID. NUM_INDICES is the number of indices and INDICES(i) contain the indices of the vector to write. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. DELTA is the number of actual indices to skip for each index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_DP` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRS,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given double precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_INTG` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRST,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given integer MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_LINTG` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRST,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)
- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_L` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRST &NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given logical MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

- subroutine `INPUT_OUTPUT::WRITE_STRING_MATRIX_SP` (ID, FIRST_ROW, DELTA_ROW, LAST_ROW, FIRST_COLUMN, DELTA_COLUMN, LAST_COLUMN, NUMBER_FIRST,&NUMBER_REPEAT, MATRIX, INDEX_FORMAT_TYPE, MATRIX_NAME_FORMAT, ROW_INDEX_FORMAT, FIRST_FORMAT, REPEAT_FORMAT, ERR, ERROR,*)

Writes the given single precision MATRIX to the given output stream specified by ID. The basic output is determined by the flag INDEX_FORMAT_TYPE. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_ONLY then the first line of output for each row is MATRIX_NAME_FORMAT concatenated named with the FIRST_FORMAT. If INDEX_FORMAT_TYPE is WRITE_STRING_MATRIX_NAME_AND_INDICES then the first line of output for each row is MATRIX_NAME_FORMAT concatenated with ROW_INDEX_FORMAT and concatenated with FIRST_FORMAT. Note that with a WRITE_STRING_MATRIX_NAME_AND_INDICES index format type the row number will be supplied to the format before the matrix data. The FIRST_FORMAT is the format initially used, followed by the REPEAT_FORMAT which is repeated as many times as necessary. NUMBER_FIRST is the number of data items in the FIRST_FORMAT and NUMBER_REPEAT is the number of data items in the REPEAT_FORMAT. FIRST_ROW/FIRST_COLUMN and LAST_ROW/LAST_COLUMN are the extents of the row/column and DELTA_ROW/DELTA_COLUMN is the NUMBER of indices to skip for each row/column index.

Variables

- INTEGER(INTG), parameter `INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_ONLY = 1`

Write the matrix name with out any indices.

- INTEGER(INTG), parameter `INPUT_OUTPUT::WRITE_STRING_MATRIX_NAME_AND_INDICES = 2`

Write the matrix name together with the matrix indices.

8.32.1 Detailed Description

Id

`input_output.f90` 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module handles all formating and input and output.

8.32.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file `input_output.f90`.

8.33 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/iso_-varying_string.f90 File Reference

Id

[iso_varying_string.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- struct [ISO_VARYING_STRING::VARYING_STRING](#)
- interface [ISO_VARYING_STRING::assignment\(=\)](#)
- interface [ISO_VARYING_STRING::adjustr](#)
- interface [ISO_VARYING_STRING::char](#)
- interface [ISO_VARYING_STRING::iachar](#)
- interface [ISO_VARYING_STRING::ichar](#)
- interface [ISO_VARYING_STRING::index](#)
- interface [ISO_VARYING_STRING::len](#)
- interface [ISO_VARYING_STRING::len_trim](#)
- interface [ISO_VARYING_STRING::lge](#)
- interface [ISO_VARYING_STRING::lgt](#)
- interface [ISO_VARYING_STRING::lle](#)
- interface [ISO_VARYING_STRING::llt](#)
- interface [ISO_VARYING_STRING::repeat](#)
- interface [ISO_VARYING_STRING::scan](#)
- interface [ISO_VARYING_STRING::trim](#)
- interface [ISO_VARYING_STRING::verify](#)
- interface [ISO_VARYING_STRING::var_str](#)
- interface [ISO_VARYING_STRING::get](#)
- interface [ISO_VARYING_STRING::put](#)
- interface [ISO_VARYING_STRING::put_line](#)
- interface [ISO_VARYING_STRING::extract](#)
- interface [ISO_VARYING_STRING::insert](#)
- interface [ISO_VARYING_STRING::remove](#)
- interface [ISO_VARYING_STRING::replace](#)
- interface [ISO_VARYING_STRING::split](#)

Namespaces

- namespace [ISO_VARYING_STRING](#)

This module provides an iso_varying_string module, conformant to the API specified in ISO/IEC 1539-2:2000 (varying-length strings for Fortran 95).

Functions

- integer [ISO_VARYING_STRING::len_\(string\)](#)
- subroutine [ISO_VARYING_STRING::op_assign_CH_VS](#) (var, exp)
- subroutine [ISO_VARYING_STRING::op_assign_VS_CH](#) (var, exp)
- type(varying_string) [ISO_VARYING_STRING::op_concat_VS_VS](#) (string_a, string_b)

- type(varying_string) ISO_VARYING_STRING::op_concat_CH_VS (string_a, string_b)
- type(varying_string) ISO_VARYING_STRING::op_concat_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_eq_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_eq_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_eq_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_ne_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_ne_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_ne_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_lt_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_lt_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_lt_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_le_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_le_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_le_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_ge_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_ge_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_ge_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::op_gt_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_gt_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::op_gt_VS_CH (string_a, string_b)
- type(varying_string) ISO_VARYING_STRING::adjustl_ (string)
- type(varying_string) ISO_VARYING_STRING::adjustr_ (string)
- function ISO_VARYING_STRING::char_auto (string)
- character(LEN=length) ISO_VARYING_STRING::char_fixed (string, length)
- integer ISO_VARYING_STRING::iachar_ (c)
- integer ISO_VARYING_STRING::ichar_ (c)
- integer ISO_VARYING_STRING::index_VS_VS (string, substring, back)
- integer ISO_VARYING_STRING::index_CH_VS (string, substring, back)
- integer ISO_VARYING_STRING::index_VS_CH (string, substring, back)
- integer ISO_VARYING_STRING::len_trim_ (string)
- logical ISO_VARYING_STRING::lge_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lge_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lge_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::lgt_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lgt_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lgt_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::lle_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lle_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::lle_VS_CH (string_a, string_b)
- logical ISO_VARYING_STRING::llt_VS_VS (string_a, string_b)
- logical ISO_VARYING_STRING::llt_CH_VS (string_a, string_b)
- logical ISO_VARYING_STRING::llt_VS_CH (string_a, string_b)
- type(varying_string) ISO_VARYING_STRING::repeat_ (string, ncopies)
- integer ISO_VARYING_STRING::scan_VS_VS (string, set, back)
- integer ISO_VARYING_STRING::scan_CH_VS (string, set, back)
- integer ISO_VARYING_STRING::scan_VS_CH (string, set, back)
- type(varying_string) ISO_VARYING_STRING::trim_ (string)
- integer ISO_VARYING_STRING::verify_VS_VS (string, set, back)
- integer ISO_VARYING_STRING::verify_CH_VS (string, set, back)
- integer ISO_VARYING_STRING::verify_VS_CH (string, set, back)

- type(varying_string) `ISO_VARYING_STRING::var_str_` (char)
- subroutine `ISO_VARYING_STRING::get_` (string, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit` (unit, string, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_set_VS` (string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_set_CH` (string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit_set_VS` (unit, string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::get_unit_set_CH` (unit, string, set, separator, maxlen, iostat)
- subroutine `ISO_VARYING_STRING::put_VS` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_CH` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_unit_VS` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_unit_CH` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_VS` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_CH` (string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_unit_VS` (unit, string, iostat)
- subroutine `ISO_VARYING_STRING::put_line_unit_CH` (unit, string, iostat)
- type(varying_string) `ISO_VARYING_STRING::extract_VS` (string, start, finish)
- type(varying_string) `ISO_VARYING_STRING::extract_CH` (string, start, finish)
- type(varying_string) `ISO_VARYING_STRING::insert_VS_VS` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::insert_CH_VS` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::insert_VS_CH` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::insert_CH_CH` (string, start, substring)
- TYPE(varying_string) `ISO_VARYING_STRING::remove_VS` (string, start, finish)
- type(varying_string) `ISO_VARYING_STRING::remove_CH` (string, start, finish)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_VS_auto` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_VS_auto` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_CH_auto` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_CH_auto` (string, start, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_VS_fixed` (string, start, finish, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_VS_fixed` (string, start, finish, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_CH_fixed` (string, start, finish, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_CH_fixed` (string, start, finish, substring)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_VS_VS_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_CH_VS_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_VS_CH_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_VS_CH_CH_target` (string, target, substring, every, back)
- type(varying_string) `ISO_VARYING_STRING::replace_CH_CH_CH_target` (string, target, substring, every, back)
- subroutine `ISO_VARYING_STRING::split_VS` (string, word, set, separator, back)
- subroutine `ISO_VARYING_STRING::split_CH` (string, word, set, separator, back)

Variables

- `integer`, parameter `ISO_VARYING_STRING::GET_BUFFER_LEN = 256`

8.33.1 Detailed Description

Id

`iso_varying_string.f90` 27 2007-07-24 16:52:51Z cpb

Author:

Rich Townsend <`rhdt@bartol.udel.edu`>

Definition in file `iso_varying_string.f90`.

8.34 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/kinds.f90 File Reference

Id

[kinds.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [KINDS](#)

This module contains all kind definitions.

Variables

- INTEGER, parameter [KINDS::INTG](#) = SELECTED_INT_KIND(9)
Standard integer kind.
- INTEGER, parameter [KINDS::SINTG](#) = SELECTED_INT_KIND(4)
Short integer kind.
- INTEGER, parameter [KINDS::LINTG](#) = SELECTED_INT_KIND(18)
Long integer kind.
- INTEGER, parameter [KINDS::PTR](#) = INTG
Pointer integer kind.
- INTEGER, parameter [KINDS::SP](#) = SELECTED_REAL_KIND(6)
Single precision real kind.
- INTEGER, parameter [KINDS::DP](#) = SELECTED_REAL_KIND(15)
Double precision real kind.
- INTEGER, parameter [KINDS::QP](#) = SELECTED_REAL_KIND(30)
Quadruple precision real kind.
- INTEGER, parameter [KINDS::SPC](#) = KIND((1.0_SP)
- INTEGER, parameter [KINDS::_SP](#)
Single precision complex kind.
- INTEGER, parameter [KINDS::DPC](#) = KIND((1.0_DP)
- INTEGER, parameter [KINDS::_DP](#)
Double precision complex kind.

8.34.1 Detailed Description

Id

[kinds.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all kind definitions.

8.34.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [kinds.f90](#).

8.35 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lapack.f90 File Reference

Id

[lapack.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [LAPACK::interface](#)

Namespaces

- namespace [LAPACK](#)

This module contains the interface descriptions to the LAPACK routines.

8.35.1 Detailed Description

Id

[lapack.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains the interface descriptions to the [LAPACK](#) routines.

8.35.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [lapack.f90](#).

8.36 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/Laplace_- equations_routines.f90 File Reference

Id

[Laplace_equations_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [LAPLACE_EQUATIONS_ROUTINES](#)

This module handles all Laplace equations routines.

Functions

- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETFINITEELEMENTCALCULATE](#) (EQUATIONS_SET, ELEMENT_NUMBER, ERR, ERROR,*)

Calculates the element stiffness matrices and RHS for a Laplace equation finite element equations set.
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETSETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the Laplace equation type of a classical field equations set class.
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETSUBTYPESET](#) (EQUATIONS_SET, EQUATIONS_SET_SUBTYPE, ERR, ERROR,*)

Sets/changes the equation subtype for a Laplace equation type of a classical field equations set class.
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_EQUATIONS_SETSTANDARDSETUP](#) (EQUATIONS_SET, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the standard Laplace equation.
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETUP](#) (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the Laplace solution.
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_SETSUBTYPESET](#) (PROBLEM, PROBLEM_SUBTYPE, ERR, ERROR,*)

Sets/changes the problem subtype for a Laplace equation type .
- subroutine [LAPLACE_EQUATIONS_ROUTINES::LAPLACE_EQUATION_PROBLEM_STANDARDSETUP](#) (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the standard Laplace equations solution.

8.36.1 Detailed Description

Id

[Laplace_equations_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all Laplace equations routines.

8.36.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [Laplace_equations_routines.f90](#).

8.37 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/lists.f90 File Reference

Id

[lists.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- struct [LISTS::LIST_PTR_TYPE](#)
Buffer type to allow arrays of pointers to a list.
- struct [LISTS::LIST_TYPE](#)
Contains information on a list.
- interface [LISTS::LIST_ITEM_ADD](#)
Adds an item to the end of a list.
- interface [LISTS::LIST_ITEM_IN_LIST](#)
Determines if an item is in a list and returns the position of the item.
- interface [LISTS::LIST_DETACH_AND_DESTROY](#)
Detaches the list values from a list and returns them as a pointer to a array of base type before destroying the list.
- interface [LISTS::LIST_SEARCH](#)
Searches a list for a given value and returns the position in the list if the value exists.
- interface [LISTS::LIST_SEARCH_LINEAR](#)
Searches a list using the linear search method.
- interface [LISTS::LIST_SORT](#)
Sorts a list into ascending order.
- interface [LISTS::LIST_SORT_BUBBLE](#)
Sorts a list into assending order using the bubble sort method.
- interface [LISTS::LIST_SORT_HEAP](#)
Sorts a list into assending order using the heap sort method.
- interface [LISTS::LIST_SORT_SHELL](#)
Sorts a list into either assending or descending order using the shell sort method.

Namespaces

- namespace [LISTS](#)
Implements lists of base types.

Functions

- subroutine [LISTS::LIST_CREATE_FINISH](#) (LIST, ERR, ERROR,*)

Finishes the creation of a list created with LIST_CREATE_START.
- subroutine [LISTS::LIST_CREATE_START](#) (LIST, ERR, ERROR,*)

Starts the creation of a list and returns a pointer to the created list.
- subroutine [LISTS::LIST_DATA_TYPE_SET](#) (LIST, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type for a list.
- subroutine [LISTS::LIST_DESTROY](#) (LIST, ERR, ERROR,*)

Destroys a list.
- subroutine [LISTS::LIST_FINALISE](#) (LIST, ERR, ERROR,*)

Finalises a list and deallocates all memory.
- subroutine [LISTS::LIST_INITIALISE](#) (LIST, ERR, ERROR,*)

Initialises a list and all its components.
- subroutine [LISTS::LIST_INITIAL_SIZE_SET](#) (LIST, INITIAL_SIZE, ERR, ERROR,*)

Sets/changes the initial size for a list.
- subroutine [LISTS::LIST_ITEM_ADD_INTG1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of an integer list.
- subroutine [LISTS::LIST_ITEM_ADD_SP1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of a single precision real list.
- subroutine [LISTS::LIST_ITEM_ADD_DP1](#) (LIST, ITEM, ERR, ERROR,*)

Adds an item to the end of a double precision real list.
- subroutine [LISTS::LIST_ITEM_IN_LIST_INTG1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given integer LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LISTS::LIST_ITEM_IN_LIST_SP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given single precision real LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LISTS::LIST_ITEM_IN_LIST_DP1](#) (LIST, ITEM, LIST_ITEM, ERR, ERROR,*)

Determines if ITEM is in the given double precision real LIST. If it is LIST_ITEM is the index in the list. If not LIST_ITEM is 0.
- subroutine [LISTS::LIST_ITEM_DELETE](#) (LIST, LIST_ITEM, ERR, ERROR,*)

Deletes the item given by the LIST_ITEM index from the given list.
- subroutine [LISTS::LIST_NUMBER_OF_ITEMS_GET](#) (LIST, NUMBER_OF_ITEMS, ERR, ERROR,*)

Gets the current number of items in a list.

- subroutine [LISTS::LIST_DETACH_AND_DESTROY_INTG](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from an integer list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

- subroutine [LISTS::LIST_DETACH_AND_DESTROY_SP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from a single precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

- subroutine [LISTS::LIST_DETACH_AND_DESTROY_DP](#) (LIST, NUMBER_IN_LIST, LIST_VALUES, ERR, ERROR,*)

Detaches the list values from a double precision real list and returns them as a pointer to a array of base type before destroying the list. The LIST_VALUES pointer must not be associated on entry. It is up to the user to then deallocate the returned list memory.

- subroutine [LISTS::LIST_REMOVE_DUPLICATES](#) (LIST, ERR, ERROR,*)

Removes duplicate entries from a list. A side effect of this is that the list is sorted.

- subroutine [LISTS::LIST_SEARCH_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches an integer array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SEARCH_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a single precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SEARCH_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a double precision real array list A for VALUE. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SEARCH_LINEAR_INTG_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches an integer array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SEARCH_LINEAR_SP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a single precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SEARCH_LINEAR_DP_ARRAY](#) (A, VALUE, POSITION, ERR, ERROR,*)

Searches a double precision real array list A for VALUE using the linear search method. If the search is successful POSITION contains the index of the position of VALUE in the list otherwise POSITION is zero.

- subroutine [LISTS::LIST_SORT_INTG_ARRAY](#) (A, ERR, ERROR,*)

Sorts an integer array list into ascending order.

- subroutine [LISTS::LIST_SORT_SP_ARRAY](#) (A, ERR, ERROR,*)

Sorts an single precision array list into ascending order.

- subroutine **LISTS::LIST_SORT_DP_ARRAY** (A, ERR, ERROR,*)

Sorts an double precision array list into ascending order.
- subroutine **LISTS::LIST_SORT_BUBBLE_INTG_ARRAY** (A, ERR, ERROR,*)

BUBBLE_SORT_INTG performs a bubble sort on an integer array list.
- subroutine **LISTS::LIST_SORT_BUBBLE_SP_ARRAY** (A, ERR, ERROR,*)

BUBBLE_SORT_SP performs a bubble sort on a single precision array list.
- subroutine **LISTS::LIST_SORT_BUBBLE_DP_ARRAY** (A, ERR, ERROR,*)

BUBBLE_SORT_DP performs a bubble sort on a double precision list.
- subroutine **LISTS::LIST_SORT_HEAP_INTG_ARRAY** (A, ERR, ERROR,*)

Sorts an integer array list into assending order using the heap sort method.
- subroutine **LISTS::LIST_SORT_HEAP_SP_ARRAY** (A, ERR, ERROR,*)

Sorts a real single precision array list into assending order using the heap sort method.
- subroutine **LISTS::LIST_SORT_HEAP_DP_ARRAY** (A, ERR, ERROR,*)

Sorts a real double precision array list into assending order using the heap sort method.
- subroutine **LISTS::LIST_SORT_SHELL_INTG_ARRAY** (A, ERR, ERROR,*)

Sorts an integer array list into either assending or descending order using the shell sort method.
- subroutine **LISTS::LIST_SORT_SHELL_SP_ARRAY** (A, ERR, ERROR,*)

Sorts a real single precision array list into either assending or descending order using the shell sort method.
- subroutine **LISTS::LIST_SORT_SHELL_DP_ARRAY** (A, ERR, ERROR,*)

Sorts a real double precision array list into either assending or descending order using the shell sort method.

Variables

- INTEGER(INTG), parameter **LISTS::LIST_INTG_TYPE** = INTEGER_TYPE

Integer data type for a list.
- INTEGER(INTG), parameter **LISTS::LIST_SP_TYPE** = SINGLE_REAL_TYPE

Single precision real data type for a list.
- INTEGER(INTG), parameter **LISTS::LIST_DP_TYPE** = DOUBLE_REAL_TYPE

Double precision real data type for a list.
- INTEGER(INTG), parameter **LISTS::LIST_UNSORTED_TYPE** = 1

Unsorted list type.
- INTEGER(INTG), parameter **LISTS::LIST_SORT_ASCENDING_TYPE** = 2

Ascending order for sort.

- INTEGER(INTG), parameter **LISTS::LIST_SORT_DESCENDING_TYPE** = 3
Descending order for sort.
- INTEGER(INTG), parameter **LISTS::LIST_BUBBLE_SORT_METHOD** = 1
Bubble sort method.
- INTEGER(INTG), parameter **LISTS::LIST_SHELL_SORT_METHOD** = 2
Shell sort method.
- INTEGER(INTG), parameter **LISTS::LIST_HEAP_SORT_METHOD** = 3
Heap sort method.

8.37.1 Detailed Description

Id

[lists.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley Implements lists of base types.

Todo

Fix up and have this module use the sorting module for sorts

8.37.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [lists.f90](#).

8.38 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_aix.f90 File Reference

Id

[machine_constants_aix.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace **MACHINE_CONSTANTS**

This module contains all machine dependent constants for AIX systems.

Variables

- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_TYPE** = IBM_COMPUTER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_OS** = AIX_OS
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_ENDIAN** = LITTLE_ENDIAN_NUMBER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_CHAR_FORMAT** = ASCII_CHARACTER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_INT_FORMAT** = TWOS_COMPLEMENT_INTEGER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_SP_FORMAT** = SPIEEE_NUMBER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::MACHINE_DP_FORMAT** = DPIEEE_NUMBER
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::INTEGER_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::SHORT_INTEGER_SIZE** = 2
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::LONG_INTEGER_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::SINGLE_REAL_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::DOUBLE_REAL_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::CHARACTER_SIZE** = 1
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::LOGICAL_SIZE** = 4
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::SINGLE_COMPLEX_SIZE** = 8
- INTEGER(INTG), parameter **MACHINE_CONSTANTS::DOUBLE_COMPLEX_SIZE** = 16
- CHARACTER(LEN=1), parameter **MACHINE_CONSTANTS::ERROR_SEPARATOR_CONSTANT** = CHAR(6)

8.38.1 Detailed Description

Id

[machine_constants_aix.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all machine dependent constants for AIX systems.

8.38.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine_constants_aix.f90](#).

8.39 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_irix.f90 File Reference

Id

[machine_constants_irix.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [MACHINE_CONSTANTS](#)

This module contains all machine dependent constants for AIX systems.

8.39.1 Detailed Description

Id

[machine_constants_irix.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all machine dependent constants for IRIX systems.

8.39.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine_constants_irix.f90](#).

8.40 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_- constants_linux.f90 File Reference

Id

[machine_constants_linux.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [MACHINE_CONSTANTS](#)

This module contains all machine dependent constants for AIX systems.

8.40.1 Detailed Description

Id

[machine_constants_linux.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all machine dependent constants for Linux systems.

8.40.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine_constants_linux.f90](#).

8.41 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_vms.f90 File Reference

Id

[machine_constants_vms.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [MACHINE_CONSTANTS](#)

This module contains all machine dependent constants for AIX systems.

8.41.1 Detailed Description

Id

[machine_constants_vms.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all machine dependent constants for VMS systems.

8.41.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine_constants_vms.f90](#).

8.42 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_- constants_win32.f90 File Reference

Id

[machine_constants_win32.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [MACHINE_CONSTANTS](#)

This module contains all machine dependent constants for AIX systems.

8.42.1 Detailed Description

Id

[machine_constants_win32.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all machine dependent constants for Win32 systems.

8.42.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [machine_constants_win32.f90](#).

8.43 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/math.f90 File Reference

Id

[math.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [MATHS::CROSS_PRODUCT](#)
- interface [MATHS::D_CROSS_PRODUCT](#)
- interface [MATHS::DETERMINANT](#)
- interface [MATHS::EDP](#)
- interface [MATHS::EIGENVALUE](#)
- interface [MATHS::EIGENVECTOR](#)
- interface [MATHS::IO](#)
- interface [MATHS::I1](#)
- interface [MATHS::INVERT](#)
- interface [MATHS::K0](#)
- interface [MATHS::K1](#)
- interface [MATHS::L2NORM](#)
- interface [MATHS::NORMALISE](#)
- interface [MATHS::SOLVE_SMALL_LINEAR_SYSTEM](#)

Namespaces

- namespace [MATHS](#)

This module contains all mathematics support routines.

Functions

- subroutine [MATHS::CROSS_PRODUCT_INTG](#) (A, B, C, ERR, ERROR,*)
- subroutine [MATHS::CROSS_PRODUCT_SP](#) (A, B, C, ERR, ERROR,*)
- subroutine [MATHS::CROSS_PRODUCT_DP](#) (A, B, C, ERR, ERROR,*)
- subroutine [MATHS::D_CROSS_PRODUCT_INTG](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [MATHS::D_CROSS_PRODUCT_SP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- subroutine [MATHS::D_CROSS_PRODUCT_DP](#) (N, A, B, C, D_A, D_B, D_C, ERR, ERROR,*)
- INTEGER(INTG) [MATHS::DETERMINANT_FULL_INTG](#) (A, ERR, ERROR)
- REAL(SP) [MATHS::DETERMINANT_FULL_SP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::DETERMINANT_FULL_DP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::EDP_DP](#) (X)
- REAL(SP) [MATHS::EDP_SP](#) (X)
- subroutine [MATHS::EIGENVALUE_FULL_SP](#) (A, EVALUES, ERR, ERROR,*)
- subroutine [MATHS::EIGENVALUE_FULL_DP](#) (A, EVALUES, ERR, ERROR,*)
- subroutine [MATHS::EIGENVECTOR_FULL_SP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)
- subroutine [MATHS::EIGENVECTOR_FULL_DP](#) (A, EVALUE, EVECTOR, ERR, ERROR,*)
- REAL(DP) [MATHS::IO_DP](#) (X)
- REAL(SP) [MATHS::IO_SP](#) (X)

- REAL(DP) [MATHS::I1_DP](#) (X)
- REAL(SP) [MATHS::I1_SP](#) (X)
- subroutine [MATHS::INVERT_FULL_SP](#) (A, B, DET, ERR, ERROR,*)
- subroutine [MATHS::INVERT_FULL_DP](#) (A, B, DET, ERR, ERROR,*)
- REAL(DP) [MATHS::K0_DP](#) (X)
- REAL(SP) [MATHS::K0_SP](#) (X)
- REAL(DP) [MATHS::K1_DP](#) (X)
- REAL(SP) [MATHS::K1_SP](#) (X)
- REAL(DP) [MATHS::KDP_DP](#) (X)
- REAL(SP) [MATHS::KDP_SP](#) (X)
- REAL(SP) [MATHS::L2NORM_SP](#) (A)
- REAL(DP) [MATHS::L2NORM_DP](#) (A)
- REAL(SP) [MATHS::NORMALISE_SP](#) (A, ERR, ERROR)
- REAL(DP) [MATHS::NORMALISE_DP](#) (A, ERR, ERROR)
- subroutine [MATHS::SOLVE_SMALL_LINEAR_SYSTEM_SP](#) (A, x, b, ERR, ERROR,*)
- subroutine [MATHS::SOLVE_SMALL_LINEAR_SYSTEM_DP](#) (A, x, b, ERR, ERROR,*)

8.43.1 Detailed Description

Id

[maths.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all mathematics support routines.

8.43.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [math.f90](#).

8.44 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/matrix_vector.f90 File Reference

Id

[matrix_vector.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- interface [MATRIX_VECTOR::MATRIX_ALL_VALUES_SET](#)
- interface [MATRIX_VECTOR::MATRIX_DATA_GET](#)
- interface [MATRIX_VECTOR::MATRIX_VALUES_ADD](#)
- interface [MATRIX_VECTOR::MATRIX_VALUES_GET](#)
- interface [MATRIX_VECTOR::MATRIX_VALUES_SET](#)
- interface [MATRIX_VECTOR::VECTOR_ALL_VALUES_SET](#)
- interface [MATRIX_VECTOR::VECTOR_DATA_GET](#)
- interface [MATRIX_VECTOR::VECTOR_VALUES_GET](#)
- interface [MATRIX_VECTOR::VECTOR_VALUES_SET](#)

Namespaces

- namespace [MATRIX_VECTOR](#)

This module contains all routines dealing with (non-distributed) matrix and vectors types.

Functions

- subroutine [MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_INTG](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in an integer matrix to the specified value.
- subroutine [MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_SP](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a single precision matrix to the specified value.
- subroutine [MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_DP](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a double precision matrix to the specified value.
- subroutine [MATRIX_VECTOR::MATRIX_ALL_VALUES_SET_L](#) (MATRIX, VALUE, ERR, ERROR,*)

Sets all values in a logical matrix to the specified value.
- subroutine [MATRIX_VECTOR::MATRIX_CREATE_FINISH](#) (MATRIX, ERR, ERROR,*)

Finishes the creation a matrix.
- subroutine [MATRIX_VECTOR::MATRIX_CREATE_START](#) (MATRIX, ERR, ERROR,*)

Starts the creation a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_INTG` (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of an integer matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_SP` (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a single precision matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_DP` (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a double precision matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

- subroutine `MATRIX_VECTOR::MATRIX_DATA_GET_L` (MATRIX, DATA, ERR, ERROR,*)

Returns a pointer to the data of a logical matrix. Note: the values can be used for read operations but a `MATRIX_VALUES_SET` call must be used to change any values. The pointer should not be deallocated.

- subroutine `MATRIX_VECTOR::MATRIX_DATA_TYPE_SET` (MATRIX, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type of a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_DESTROY` (MATRIX, ERR, ERROR,*)

Destroys a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_DUPLICATE` (MATRIX, NEW_MATRIX, ERR, ERROR,*)

Duplicates the matrix and returns a pointer to the duplicated matrix in NEWMATRIX.

- subroutine `MATRIX_VECTOR::MATRIX_FINALISE` (MATRIX, ERR, ERROR,*)

Finalises a matrix and deallocates all memory.

- subroutine `MATRIX_VECTOR::MATRIX_INITIALISE` (MATRIX, ERR, ERROR,*)

Initialises a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_MAX_COLUMNS_PER_ROW_GET` (MATRIX, MAX_COLUMNS_PER_ROW, ERR, ERROR,*)

Gets the maximum number of columns in each row of a distributed matrix.

- subroutine `MATRIX_VECTOR::MATRIX_NUMBER_NON_ZEROS_SET` (MATRIX, NUMBER_NON_ZEROS, ERR, ERROR,*)

Sets/changes the number of non zeros for a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_MAX_SIZE_SET` (MATRIX, MAX_M, MAX_N, ERR, ERROR,*)

Sets/changes the maximum size of a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_OUTPUT` (ID, MATRIX, ERR, ERROR,*)

Sets/changes the size of a matrix.

- subroutine `MATRIX_VECTOR::MATRIX_SIZE_SET` (MATRIX, M, N, ERR, ERROR,*)

Sets/changes the size of a matrix.

- subroutine [MATRIX_VECTOR::MATRIX_STORAGE_LOCATION_FIND](#) (MATRIX, I, J, LOCATION, ERR, ERROR,*)

Returns the storage location in the data array of a matrix that corresponds to location I,J. If the location does not exist the routine returns zero.

- subroutine [MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_GET](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)

Gets the storage locations (sparsity pattern) of a matrix.

- subroutine [MATRIX_VECTOR::MATRIX_STORAGE_LOCATIONS_SET](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)

Sets the storage locations (sparsity pattern) in a matrix to that specified by the row and column indices.

- subroutine [MATRIX_VECTOR::MATRIX_STORAGE_TYPE_GET](#) (MATRIX, STORAGE_TYPE, ERR, ERROR,*)

Gets the storage type for a matrix.

- subroutine [MATRIX_VECTOR::MATRIX_STORAGE_TYPE_SET](#) (MATRIX, STORAGE_TYPE, ERR, ERROR,*)

Sets/changes the storage type for a matrix.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_INTG2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_SP](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_SP1](#) (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a single precision real matrix at the location specified by the row and column index i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine [MATRIX_VECTOR::MATRIX_VALUES_ADD_SP2](#) (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a single precision real matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=MATRIX(I,J)+VALUE.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_DP** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_DP1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a double precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_DP2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J)+\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_L** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds values to a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_L1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Adds a value to a logical matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_ADD_L2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Adds a matrix of values to a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{MATRIX}(I,J).\text{OR}.\text{VALUE}$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_INTG** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets the values in an integer matrix at the location specified by the row and column indices i.e., $\text{VALUE}=\text{MATRIX}(I,J)$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_INTG1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in an integer matrix at the location specified by the row and column index i.e., $\text{VALUE}=\text{MATRIX}(I,J)$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_INTG2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., $\text{VALUE}=\text{MATRIX}(I,J)$.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_SP** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{VALUE}=\text{MATRIX}(I,J)$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_SP1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a single precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_SP2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_DP** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_DP1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a double precision real matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_DP2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_L** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_L1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Gets a value in a logical matrix at the location specified by the row and column index i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_GET_L2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Gets a matrix of values in a logical matrix at the location specified by the row and column indices i.e., VALUE=MATRIX(I,J).
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_INTG** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_INTG1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in an integer matrix at the location specified by the row and column index i.e., MATRIX(I,J)=VALUE.
- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_INTG2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets a matrix of values in an integer matrix at the location specified by the row and column indices i.e., MATRIX(I,J)=VALUE.

Sets the matrix of values in an integer matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_SP** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_SP1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets the value in a single precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_SP2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a single precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_DP** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_DP1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in a double precision real matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_DP2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a double precision real matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_L** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the values in a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_L1** (MATRIX, ROW_INDEX, COLUMN_INDEX, VALUE, ERR, ERROR,*)

Sets a value in a logical matrix at the location specified by the row and column index i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::MATRIX_VALUES_SET_L2** (MATRIX, ROW_INDICES, COLUMN_INDICES, VALUES, ERR, ERROR,*)

Sets the matrix of values in a logical matrix at the location specified by the row and column indices i.e., $\text{MATRIX}(I,J)=\text{VALUE}$.

- subroutine **MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_INTG** (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in an integer vector to the specified value.

- subroutine [MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_SP](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a single precision vector to the specified value.
- subroutine [MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_DP](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a double precision vector to the specified value.
- subroutine [MATRIX_VECTOR::VECTOR_ALL_VALUES_SET_L](#) (VECTOR, VALUE, ERR, ERROR,*)

Sets all values in a logical vector to the specified value.
- subroutine [MATRIX_VECTOR::VECTOR_CREATE_FINISH](#) (VECTOR, ERR, ERROR,*)

Finishes the creation of a vector.
- subroutine [MATRIX_VECTOR::VECTOR_CREATE_START](#) (VECTOR, ERR, ERROR,*)

Starts the creation a vector.
- subroutine [MATRIX_VECTOR::VECTOR_DATA_GET_INTG](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of an integer vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_VECTOR::VECTOR_DATA_GET_SP](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a single precision vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_VECTOR::VECTOR_DATA_GET_DP](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a double precision vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_VECTOR::VECTOR_DATA_GET_L](#) (VECTOR, DATA, ERR, ERROR,*)

Returns a pointer to the data of a logical vector. Note: the values can be used for read operations but a [VECTOR_VALUES_SET](#) call must be used to change any values. The pointer should not be deallocated.
- subroutine [MATRIX_VECTOR::VECTOR_DATA_TYPE_SET](#) (VECTOR, DATA_TYPE, ERR, ERROR,*)

Sets/changes the data type of a vector.
- subroutine [MATRIX_VECTOR::VECTOR_DESTROY](#) (VECTOR, ERR, ERROR,*)

Destroys a vector.
- subroutine [MATRIX_VECTOR::VECTOR_DUPLICATE](#) (VECTOR, NEW_VECTOR, ERR, ERROR,*)

Duplicates a vector structure and returns a pointer to the new vector in NEW_VECTOR.
- subroutine [MATRIX_VECTOR::VECTOR_FINALISE](#) (VECTOR, ERR, ERROR,*)

Finalises a vector and deallocates all memory.

- subroutine [MATRIX_VECTOR::VECTOR_INITIALISE](#) (VECTOR, ERR, ERROR,*)

Initialises a vector.
- subroutine [MATRIX_VECTOR::VECTOR_SIZE_SET](#) (VECTOR, N, ERR, ERROR,*)

Sets/changes the size of a vector.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in an integer vector at the indices specified.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in an integer vector at the location specified by the index.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a single precision real vector at the indices specified.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_SP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a single precision vector at the location specified by the index.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_DP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a double precision real vector at the indices specified.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_DP1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a double precision vector at the location specified by the index.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_L](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Gets the values in a logical real vector at the indices specified.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_GET_L1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Gets a value in a logical vector at the location specified by the index.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_SET_INTG](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in an integer vector at the specified indices.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_SET_INTG1](#) (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in an integer vector at the specified index.
- subroutine [MATRIX_VECTOR::VECTOR_VALUES_SET_SP](#) (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a single precision vector at the specified indices.

- subroutine `MATRIX_VECTOR::VECTOR_VALUES_SET_SP1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a single precision vector at the specified index.
- subroutine `MATRIX_VECTOR::VECTOR_VALUES_SET_DP` (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a double precision vector at the specified indices.
- subroutine `MATRIX_VECTOR::VECTOR_VALUES_SET_DPI` (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a double precision vector at the specified index.
- subroutine `MATRIX_VECTOR::VECTOR_VALUES_SET_L` (VECTOR, INDICES, VALUES, ERR, ERROR,*)

Sets the values in a logical vector at the specified indices.
- subroutine `MATRIX_VECTOR::VECTOR_VALUES_SET_L1` (VECTOR, INDEX, VALUE, ERR, ERROR,*)

Sets a value in a logical vector at the specified index.

Variables

- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_VECTOR_INTG_TYPE` = `INTEGER_TYPE`

Integer matrix-vector data type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_VECTOR_SP_TYPE` = `SINGLE_REAL_TYPE`

Single precision real matrix-vector data type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_VECTOR_DP_TYPE` = `DOUBLE_REAL_TYPE`

Double precision real matrix-vector data type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_VECTOR_L_TYPE` = `LOGICAL_TYPE`

Logical matrix-vector data type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_BLOCK_STORAGE_TYPE` = 0

Matrix block storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_DIAGONAL_STORAGE_TYPE` = 1

Matrix diagonal storage type.
- INTEGER(INTG), parameter `MATRIX_VECTOR::MATRIX_COLUMN_MAJOR_STORAGE_TYPE` = 2

Matrix column major storage type.

- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_ROW_MAJOR_STORAGE_TYPE** = 3
Matrix row major storage type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_COMPRESSED_ROW_STORAGE_TYPE** = 4
Matrix compressed row storage type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE** = 5
Matrix compressed column storage type.
- INTEGER(INTG), parameter **MATRIX_VECTOR::MATRIX_ROW_COLUMN_STORAGE_TYPE** = 6
Matrix row-column storage type.
- INTEGER(INTG), save **MATRIX_VECTOR::MATRIX_VECTOR_ID** = 1

8.44.1 Detailed Description

Id

[matrix_vector.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module contains all routines dealing with (non-distributed) matrix and vectors types.

8.44.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL

or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

8.44.3 MATRIX STORAGE STRUCTURES

The matrix storage structures used are governed by the STORAGE parameter associated with the array. If STORAGE is MATRIX_BLOCK_STORAGE_TYPE the matrix is not sparse and the non-sparse matrix dimension M is used to calculate the matrix storage locations. If storage is MATRIX_DIAGONAL_STORAGE_TYPE then only the matrix diagonal is stored. If storage is MATRIX_COLUMN_MAJOR_STORAGE_TYPE the matrix is not sparse and the non-sparse matrix dimension MAX_M ($\geq M$) is used to calculate the matrix storage locations. If storage is MATRIX_ROW_MAJOR_STORAGE_TYPE the matrix is not sparse and the non-sparse matrix dimension MAX_N ($\geq N$) is used to calculate the matrix storage locations. If STORAGE is MATRIX_COMPRESSED_ROW_STORAGE_TYPE the matrix has compressed row storage/sparsity (see below) and the sparsity structure arrays ROW_INDICES and COLUMN_INDICES are used for the storage location calculation. If STORAGE is MATRIX_COMPRESSED_COLUMN_STORAGE_TYPE the matrix has compressed column storage/sparsity (see below) and the sparsity structure arrays ROW_INDICES and COLUMN_INDICES are used for the storage location calculation. If STORAGE is MATRIX_ROW_COLUMN_STORAGE_TYPE the matrix has row column storage/sparsity (see below) and the sparsity structure arrays ROW_INDICES and COLUMN_INDICES are used for the storage location calculation.

8.44.3.1 COMPRESSED-ROW STORAGE:

The storage structure scheme is based on storing a $M \times N$ matrix as a one dimensional array of length SIZE (=NUMBER_NON_ZEROS) (where NUMBER_NON_ZEROS= $s \times M \times N$, s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW_INDICES and COLUMN_INDICES store the positions of the non-zero elements. ROW_INDICES is of length $M+1$ and COLUMN is of length NUMBER_NON_ZEROS. ROW_INDICES(i) stores the position in COLUMN_INDICES of the start of row i. The $M+1$ position of ROW_INDICES stores the size of COLUMN_INDICES+1 i.e., NUMBER_NON_ZEROS+1. The number of non-zero elements in row i can be found from ROW_INDICES(i+1)-ROW_INDICES(i). COLUMN_INDICES(nz) gives the column number for non-zero element nz. See also COMPRESSED-COLUMN storage.

Example of the compressed-row storage scheme on a $N \times N$ matrix ($N=6$). Here the sparsity is 8/36 or 22%

GX	1	2	3	4	5	6	
1	0	A	0	B	0	0	
2	0	0	C	0	0	0	
3	0	0	0	0	D	E	
4	F	0	0	0	0	0	
5	0	0	G	0	0	0	
6	0	0	0	0	0	H	

DATA (nz)							
A	B	C	D	E	F	G	H
ROW_INDICES (i)							
1	3	4	6	7	8	9	
COLUMN_INDICES (i)							
2	4	3	5	6	1	3	6

8.44.3.2 COMPRESSED-COLUMN STORAGE:

The storage structure scheme is based on storing a $M \times N$ matrix as a one dimensional array of length SIZE (=NUMBER_NON_ZEROS) (where NUMBER_NON_ZEROS= $s \times M \times N$, s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW_INDICES and COLUMN_INDICES store the positions of the non-zero elements. ROW_INDICES is of length NUMBER_NON_ZEROS and COLUMN is of length $N+1$. COLUMN_INDICES(j) stores the position in ROW_INDICES

of the start of column j. The N+1 position of COLUMN_INDICES stores the size of ROW_INDICES+1 i.e., NUMBER_NON_ZEROS+1. The number of non-zero elements in column j can be found from COLUMN_INDICES(j+1)-COLUMN_INDICES(j). ROW_INDICES(nz) gives the row number for non-zero element nz. See also COMPRESSED-ROW storage.

Example of compressed-column storage scheme on a NxN matrix (N=6). Here the sparsity is 8/36 or 22%

GX 1 2 3 4 5 6	
<hr/>	
1 0 A 0 B 0 0	DATA (nz)
2 0 0 C 0 0 0	F A C G B D E H
3 0 0 0 0 D E	ROW_INDICES (i)
4 F 0 0 0 0 0	4 1 2 5 1 3 3 6
5 0 0 G 0 0 0	COLUMN_INDICES (i)
6 0 0 0 0 0 H	1 2 3 5 6 7 9

8.44.3.3 ROW-COLUMN STORAGE:

The storage structure scheme is based on storing a MxN matrix as a one dimensional array of length SIZE (=NUMBER_NON_ZEROS) (where NUMBER_NON_ZEROS=sxMxN, s is the sparsity of the array) that stores only the non-zero elements of the matrix. Two additional arrays ROW_INDICES and COLUMN_INDICES store the positions of the non-zero elements. Both ROW_INDICES and COLUMN_INDICES are of length NUMBER_NON_ZEROS. ROW_INDICES(nz) gives the row number for non-zero element nz and COLUMN_INDICES(nz) gives the column number for non-zero element nz.

Example of row-column storage scheme on a NxN matrix (N=6). Here the sparsity is 8/36 or 22%

GX 1 2 3 4 5 6	
<hr/>	
1 0 A 0 B 0 0	DATA (nz)
2 0 0 C 0 0 0	A B C D E F G H
3 0 0 0 0 D E	ROW_INDICES (i)
4 F 0 0 0 0 0	1 1 2 3 3 4 5 6
5 0 0 G 0 0 0	COLUMN_INDICES (i)
6 0 0 0 0 0 H	2 4 3 5 6 1 3 6

Definition in file [matrix_vector.f90](#).

8.45 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/mesh_routines.f90 File Reference

Id

[mesh_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- interface [MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET](#)
Sets/changes the number of mesh components for a mesh.
- interface [MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET](#)
Sets/changes the number of elements for a mesh.

Namespaces

- namespace [MESH_ROUTINES](#)
This module handles all mesh (node and element) routines.

Functions

- subroutine [MESH_ROUTINES::DECOMPOSITION_CREATE_FINISH](#) (MESH, DECOMPOSITION, ERR, ERROR,*)
Finishes the creation of a domain decomposition on a given mesh.
- subroutine [MESH_ROUTINES::DECOMPOSITION_CREATE_START](#) (USER_NUMBER, MESH, DECOMPOSITION, ERR, ERROR,*)
Starts the creation of a domain decomposition for a given mesh.
- subroutine [MESH_ROUTINES::DECOMPOSITION_DESTROY](#) (USER_NUMBER, MESH, ERR, ERROR,*)
Destroys a domain decomposition identified by a user number and deallocates all memory.
- subroutine [MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_CALCULATE](#) (DECOMPOSITION, ERR, ERROR,*)
Calculates the element domains for a decomposition of a mesh.
- subroutine [MESH_ROUTINES::DECOMPOSITION_ELEMENT_DOMAIN_SET](#) (DECOMPOSITION, GLOBAL_ELEMENT_NUMBER, DOMAIN_NUMBER, ERR, ERROR,*)
Sets the domain for a given element in a decomposition of a mesh.
- subroutine [MESH_ROUTINES::DECOMPOSITION_MESH_COMPONENT_NUMBER_SET](#) (DECOMPOSITION, MESH_COMPONENT_NUMBER, ERR, ERROR,*)
Sets/changes the mesh component number which will be used for the decomposition of a mesh.
- subroutine [MESH_ROUTINES::DECOMPOSITION_NUMBER_OF_DOMAINS_SET](#) (DECOMPOSITION, NUMBER_OF_DOMAINS, ERR, ERROR,*)

Sets/changes the number of domains for a decomposition.

- subroutine `MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_FINALISE` (DOMAIN_-
MAPPINGS, ERR, ERROR,*)

Finalises the node mapping in the given domain mappings.
- subroutine `MESH_ROUTINES::DOMAIN_MAPPINGS_NODES_INITIALISE` (DOMAIN_-
MAPPINGS, ERR, ERROR,*)

Initialises the node mapping in the given domain mapping.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_CALCULATE` (TOPOLOGY, ERR,
ERROR,*)

Calculates the domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE_FROM_MESH` (DO-
MAIN, ERR, ERROR,*)

Initialises the local domain topology from the mesh topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_FINALISE` (TOPOLOGY, ERR,
ERROR,*)

Finalises the dofs in the given domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_DOFS_INITIALISE` (TOPOLOGY,
ERR, ERROR,*)

Initialises the dofs data structures for a domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENT_FINALISE` (ELEMENT,
ERR, ERROR,*)

Finalises the given domain topology element.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENT_INITIALISE` (ELEMENT,
ERR, ERROR,*)

Initialises the given domain topology element.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_FINALISE` (TOPOLOGY,
ERR, ERROR,*)

Finalises the elements in the given domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_ELEMENTS_INITIALISE` (TOPOL-
OGY, ERR, ERROR,*)

Initialises the element data structures for a domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_FINALISE` (DOMAIN, ERR,
ERROR,*)

Finalises the topology in the given domain.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_INITIALISE` (DOMAIN, ERR,
ERROR,*)

Initialises the topology for a given domain.

- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_FINALISE` (LINE, ERR, ERROR,*)

Finalises a line in the given domain topology and deallocates all memory.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_LINE_INITIALISE` (LINE, ERR, ERROR,*)

Initialises the line data structure for a domain topology line.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_FINALISE` (TOPOLOGY, ERR, ERROR,*)

Finalises the lines in the given domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_LINES_INITIALISE` (TOPOLOGY, ERR, ERROR,*)

Initialises the line data structures for a domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_FINALISE` (NODE, ERR, ERROR,*)

Finalises the given domain topology node and deallocates all memory.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODE_INITIALISE` (NODE, ERR, ERROR,*)

Initialises the given domain topology node.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_FINALISE` (TOPOLOGY, ERR, ERROR,*)

Finalises the nodees in the given domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_INITIALISE` (TOPOLOGY, ERR, ERROR,*)

Initialises the nodes data structures for a domain topology.
- subroutine `MESH_ROUTINES::DOMAIN_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_CALCULATE` (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding a node for a domain.
- subroutine `MESH_ROUTINES::MESH_CREATE_FINISH` (REGION, MESH, ERR, ERROR,*)

Finishes the process of creating a mesh on a region.
- subroutine `MESH_ROUTINES::MESH_CREATE_REGULAR` (USER_NUMBER, REGION, ORIGIN, MAXIMUM_EXTENT, NUMBER_ELEMENTS_XI, BASIS, MESH, ERR, ERROR,*)

Creates the regular mesh with the given USER_NUMBER in the specified REGION. The mesh starts at the ORIGIN(:) and has a maximum extent position of MAXIMUM_EXTENT(:) with the NUMBER_OF_ELEMENTS(:) in each direction. Each element is of the specified BASIS type. A pointer to the finished mesh is returned in MESH.
- subroutine `MESH_ROUTINES::MESH_CREATE_START` (USER_NUMBER, REGION, NUMBER_OF_DIMENSIONS, MESH, ERR, ERROR,*)

Starts the process of creating a mesh defined by a user number with the specified NUMBER_OF_DIMENSIONS in the region identified by REGION.

- subroutine **MESH_ROUTINES::MESH_DESTROY** (USER_NUMBER, REGION, ERR, ERROR,*)

Destroys the mesh identified by a user number on the given region and deallocates all memory.
- subroutine **MESH_ROUTINES::MESH_FINALISE** (MESH, ERR, ERROR,*)

Finalises a mesh and deallocates all memory.
- subroutine **MESH_ROUTINES::MESH_INITIALISE** (MESH, ERR, ERROR,*)

Initialises a mesh.
- subroutine **MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_NUMBER** (USER_NUMBER, REGION, NUMBER_OF_COMPONENTS, ERR, ERROR,*)

Changes/sets the number of mesh components for a mesh identified by a given user number on a region.
- subroutine **MESH_ROUTINES::MESH_NUMBER_OF_COMPONENTS_SET_PTR** (MESH, NUMBER_OF_COMPONENTS, ERR, ERROR,*)

Changes/sets the number of mesh components for a mesh identified by a pointer.
- subroutine **MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_NUMBER** (USER_NUMBER, REGION, NUMBER_OF_ELEMENTS, ERR, ERROR,*)

Changes/sets the number of elements for a mesh identified by a given user number on a region.
- subroutine **MESH_ROUTINES::MESH_NUMBER_OF_ELEMENTS_SET_PTR** (MESH, NUMBER_OF_ELEMENTS, ERR, ERROR,*)

Changes/sets the number of elements for a mesh identified by a pointer.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_CALCULATE** (TOPOLOGY, ERR, ERROR,*)

Calculates the mesh topology.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_CALCULATE** (TOPOLOGY, ERR, ERROR,*)

Calculates the degrees-of-freedom for a mesh topology.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_FINALISE** (TOPOLOGY, ERR, ERROR,*)

Finalises the dof data structures for a mesh topology and deallocates any memory.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_DOFS_INITIALISE** (TOPOLOGY, ERR, ERROR,*)

Initialises the dofs in a given mesh topology.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_BASIS_SET** (GLOBAL_NUMBER, ELEMENTS, BASIS, ERR, ERROR,*)

Changes/sets the basis for a global element identified by a given global number.
- subroutine **MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_FINISH** (MESH, MESH_COMPONENT_NUMBER, ERR, ERROR,*)

Finishes the process of creating elements for a specified mesh component in a mesh topology.

- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_CREATE_START` (MESH, MESH_COMPONENT_NUMBER, BASIS, ELEMENTS, ERR, ERROR,*)

Starts the process of creating elements in the mesh component identified by MESH and component_idx. The elements will be created with a default basis of BASIS. ELEMENTS is the returned pointer to the MESH-ELEMENTS data structure.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_DESTROY` (TOPOLOGY, ERR, ERROR,*)

Destroys the elements in a mesh topology.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_FINALISE` (ELEMENT, ERR, ERROR,*)

Finalises the given mesh topology element.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENT_INITIALISE` (ELEMENT, ERR, ERROR,*)

Initialises the given mesh topology element.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_BASIS_SET` (GLOBAL_NUMBER, ELEMENTS, BASIS, ERR, ERROR,*)

Changes/sets the basis for a mesh element identified by a given global number.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ELEMENT_NODES_SET` (GLOBAL_NUMBER, ELEMENTS, USER_ELEMENT_NODES, ERR, ERROR,*)

Changes/sets the element nodes for a mesh element identified by a given global number.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_ADJACENT_ELEMENTS_CALCULATE` (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding an element in a mesh topology.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_FINALISE` (TOPOLOGY, ERR, ERROR,*)

Finalises the elements data structures for a mesh topology and deallocates any memory.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_INITIALISE` (TOPOLOGY, ERR, ERROR,*)

Initialises the elements in a given mesh topology.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_ELEMENTS_NUMBER_SET` (GLOBAL_NUMBER, USER_NUMBER, ELEMENTS, ERR, ERROR,*)

Changes/sets the user number for a global element identified by a given global number.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_FINALISE` (MESH, ERR, ERROR,*)

Finalises the topology in the given mesh.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_INITIALISE` (MESH, ERR, ERROR,*)

Initialises the topology for a given mesh.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODE_FINALISE` (NODE, ERR, ERROR,*)

Finalises the given mesh topology node.

- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODE_INITIALISE` (NODE, ERR, ERROR,*)

Initialises the given mesh topology node.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODES_CALCULATE` (TOPOLOGY, ERR, ERROR,*)

Calculates the nodes used the mesh identified by a given mesh topology.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODES_DERIVATIVES_CALCULATE` (TOPOLOGY, ERR, ERROR,*)

Calculates the number of derivatives at each node in a topology.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODES_SURROUNDING_ELEMENTS_-_CALCULATE` (TOPOLOGY, ERR, ERROR,*)

Calculates the element numbers surrounding a node for a mesh.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODES_FINALISE` (TOPOLOGY, ERR, ERROR,*)

Finalises the nodes data structures for a mesh topology and deallocates any memory.
- subroutine `MESH_ROUTINES::MESH_TOPOLOGY_NODES_INITIALISE` (TOPOLOGY, ERR, ERROR,*)

Initialises the nodes in a given mesh topology.
- subroutine `MESH_ROUTINES::MESH_USER_NUMBER_FIND` (USER_NUMBER, REGION, MESH, ERR, ERROR,*)

Finds and returns in MESH a pointer to the mesh identified by USER_NUMBER in the given REGION. If no mesh with that number exists MESH is left nullified.
- subroutine `MESH_ROUTINES::MESSES_FINALISE` (REGION, ERR, ERROR,*)

Finalises the meshes in the given region.
- subroutine `MESH_ROUTINES::MESSES_INITIALISE` (REGION, ERR, ERROR,*)

Initialises the meshes for the given region.

Variables

- INTEGER(INTG), parameter `MESH_ROUTINES::DECOMPOSITION_ALL_TYPE = 1`

The decomposition contains all elements.
- INTEGER(INTG), parameter `MESH_ROUTINES::DECOMPOSITION_CALCULATED_TYPE = 2`

The element decomposition is calculated by graph partitioning.
- INTEGER(INTG), parameter `MESH_ROUTINES::DECOMPOSITION_USER_DEFINED_TYPE = 3`

The user will set the element decomposition.

8.45.1 Detailed Description

Id

[mesh_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all mesh (node and element) routines.

8.45.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [mesh_routines.f90](#).

8.46 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/node_routines.f90 File Reference

Id

[node_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Namespaces

- namespace [NODE_ROUTINES](#)

This module handles all node routines.

Functions

- subroutine [NODE_ROUTINES::NODE_CHECK_EXISTS](#) (USER_NUMBER, REGION, NODE_EXISTS, GLOBAL_NUMBER, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODE_DESTROY](#) (NODE, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODE_INITIAL_POSITION_SET](#) (GLOBAL_NUMBER, INITIAL_POSITION, NODES, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODE_NUMBER_SET](#) (GLOBAL_NUMBER, USER_NUMBER, NODES, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODES_CREATE_FINISH](#) (REGION, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODES_CREATE_START](#) (NUMBER_OF_NODES, REGION, NODES, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODES_FINALISE](#) (REGION, ERR, ERROR,*)
- subroutine [NODE_ROUTINES::NODES_INITIALISE](#) (REGION, ERR, ERROR,*)

8.46.1 Detailed Description

Id

[node_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module handles all node routines.

8.46.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [node_routines.f90](#).

8.47 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_constants.f90 File Reference

Id

[problem_constants.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace PROBLEM_CONSTANTS

This module handles all problem wide constants.

Variables

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_NO_CLASS = 0
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_ELASTICITY_CLASS = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_FLUID_MECHANICS_CLASS = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_ELECTROMAGNETICS_CLASS = 3
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_CLASSICAL_FIELD_CLASS = 4
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_MODAL_CLASS = 5
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_FITTING_CLASS = 6
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_OPTIMISATION_CLASS = 7
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_NO_TYPE = 0
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_LINEAR_ELASTICITY_TYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEMFINITE_ELASTICITY_TYPE = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_STOKES_FLUID_TYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_NAVIER_STOKES_FLUID_TYPE = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_ELECTROSTATIC_TYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_MAGNETOSTATIC_TYPE = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_MAXWELLS_EQUATIONS_TYPE = 3
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_LAPLACE_EQUATION_TYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_POISSON_EQUATION_TYPE = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_HELMHOLTZ_EQUATION_TYPE = 3
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_WAVE_EQUATION_TYPE = 4

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_DIFFUSION_-
EQUATION_TYPE = 5
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_ADVECTION_-
DIFFUSION_EQUATION_TYPE = 6
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_REACTION_-
DIFFUSION_EQUATION_TYPE = 7
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_BIHAMONIC_-
EQUATION_TYPE = 8
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_LINEAR_ELASTIC_-
MODAL_TYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_NO_SUBTYPE = 0
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_STANDARD_LAPLACE_-
SUBTYPE = 1
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_GENERALISED_-
LAPLACE_SUBTYPE = 2
- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_INITIAL_TYPE = 1

Initial setup for a problem.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_CONTROL_TYPE = 2

Solver setup for a problem.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLUTION_-
TYPE = 3

Solution parameters setup for a problem.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_SOLVER_TYPE = 4

Solver setup for a problem.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_START_ACTION = 1

Start setup action.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_FINISH_ACTION = 2

Finish setup action.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM_SETUP_DO_ACTION = 3

Do setup action.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_NO_-
OUTPUT = 0

No output.

- INTEGER(INTG), parameter PROBLEM_CONSTANTS::PROBLEM SOLUTION_TIMING_-
OUTPUT = 1

Timing information output.

- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLUTION MATRIX - OUTPUT = 2**

All below and solution matrices output.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLVER NO_OUTPUT = 0**

No output.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLVER TIMING - OUTPUT = 1**

Timing information output.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLVER SOLVER - OUTPUT = 2**

All below and solver output.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLVER SPARSE - MATRICES = 1**

Use sparse solver matrices.
- INTEGER(INTG), parameter **PROBLEM_CONSTANTS::PROBLEM SOLVER FULL - MATRICES = 2**

Use fully populated solver matrices.

8.47.1 Detailed Description

Id

[problem_constants.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all problem wide constants.

8.47.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [problem_constants.f90](#).

8.48 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_routines.f90 File Reference

Id

[problem_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- interface [PROBLEM_ROUTINES::PROBLEM_DESTROY](#)
- interface [PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET](#)

Namespaces

- namespace [PROBLEM_ROUTINES](#)

This module handles all problem routines.

Functions

- subroutine [PROBLEM_ROUTINES::PROBLEM_CREATE_FINISH](#) (PROBLEM, ERR, ERROR,*)
Finishes the process of creating a problem.
- subroutine [PROBLEM_ROUTINES::PROBLEM_CREATE_START](#) (USER_NUMBER, PROBLEM, ERR, ERROR,*)
Starts the process of creating a problem defined by USER_NUMBER.
- subroutine [PROBLEM_ROUTINES::PROBLEM_DESTROY_NUMBER](#) (USER_NUMBER, ERR, ERROR,*)
Destroys a problem identified by a user number.
- subroutine [PROBLEM_ROUTINES::PROBLEM_DESTROY_PTR](#) (PROBLEM, ERR, ERROR,*)
Destroys a problem identified by a pointer.
- subroutine [PROBLEM_ROUTINES::PROBLEM_FINALISE](#) (PROBLEM, ERR, ERROR,*)
Finalise the problem and deallocate all memory.
- subroutine [PROBLEM_ROUTINES::PROBLEM_INITIALISE](#) (PROBLEM, ERR, ERROR,*)
Initialises a problem.
- subroutine [PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_FINISH](#) (PROBLEM, ERR, ERROR,*)
Finish the creation of the control for the problem.
- subroutine [PROBLEM_ROUTINES::PROBLEM_CONTROL_CREATE_START](#) (PROBLEM, ERR, ERROR,*)
Start the creation of a problem control for a problem.

- subroutine **PROBLEM_ROUTINES::PROBLEM_CONTROL_DESTROY** (PROBLEM, ERR, ERROR,*)

Destroy the control for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_CONTROL_FINALISE** (CONTROL, ERR, ERROR,*)

Finalise the control for a problem and deallocate all memory.
- subroutine **PROBLEM_ROUTINES::PROBLEM_CONTROL_INITIALISE** (PROBLEM, ERR, ERROR,*)

Initialise the control for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SETUP** (PROBLEM, SETUP_TYPE, ACTION_TYPE, ERR, ERROR,*)

Sets up the specifics for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SOLVE** (PROBLEM, ERR, ERROR,*)

Solves a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTION_SOLVE** (SOLUTION, ERR, ERROR,*)

Solves a solution.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTIONS_CREATE_FINISH** (PROBLEM, ERR, ERROR,*)

Finish the creation of solutions for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTIONS_CREATE_START** (PROBLEM, ERR, ERROR,*)

Start the creation of a solution for the problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTION_EQUATIONS_SET_ADD** (PROBLEM, SOLUTION_INDEX, EQUATIONS_SET, EQUATIONS_SET_INDEX, ERR, ERROR,*)

Adds an equations set to a problem solution.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTION_FINALISE** (SOLUTION, ERR, ERROR,*)

Finalises a solution and deallocates all memory.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTION_INITIALISE** (SOLUTION, ERR, ERROR,*)

Initialises the solution for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTIONS_FINALISE** (PROBLEM, ERR, ERROR,*)

Finalises the solutions for a problem and deallocates all memory.
- subroutine **PROBLEM_ROUTINES::PROBLEM SOLUTIONS_INITIALISE** (PROBLEM, ERR, ERROR,*)

Initialises the solutions for a problem.

- subroutine **PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_FINISH** (PROBLEM, ERR, ERROR,*)
Finish the creation of the solver for the problem solutions.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SOLVER_CREATE_START** (PROBLEM, ERR, ERROR,*)
Start the creation of a problem solver for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SOLVER_DESTROY** (PROBLEM, ERR, ERROR,*)
Destroy the solvers for a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SOLVER_GET** (PROBLEM, SOLUTION_INDEX, SOLVER, ERR, ERROR,*)
Returns a pointer to the solver for a solution on a problem.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_NUMBER** (USER_NUMBER, PROBLEM_CLASS, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)
Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a user number.
- subroutine **PROBLEM_ROUTINES::PROBLEM_SPECIFICATION_SET_PTR** (PROBLEM, PROBLEM_CLASS, PROBLEM_EQUATION_TYPE, PROBLEM_SUBTYPE, ERR, ERROR,*)
Sets/changes the problem specification i.e., problem class, type and subtype for a problem identified by a pointer.
- subroutine **PROBLEM_ROUTINES::PROBLEM_USER_NUMBER_FIND** (USER_NUMBER, PROBLEM, ERR, ERROR,*)
Finds and returns in PROBLEM a pointer to the problem identified by USER_NUMBER. If no problem with that USER_NUMBER exists PROBLEM is left nullified.
- subroutine **PROBLEM_ROUTINES::PROBLEMS_FINALISE** (ERR, ERROR,*)
Finalises all problems and deallocates all memory.
- subroutine **PROBLEM_ROUTINES::PROBLEMS_INITIALISE** (ERR, ERROR,*)
Initialises all problems.

Variables

- INTEGER(INTG), parameter **PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_LINEARITIES** = 3
The number of problem linearity types defined.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_LINEAR** = 1
The problem is linear.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_NONLINEAR** = 2
The problem is non-linear.

- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_NONLINEAR_BCS** = 3
The problem has non-linear boundary conditions.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::NUMBER_OF_PROBLEM_TIME_TYPES** = 3
The number of problem time dependence types defined.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_STATIC** = 1
The problem is static and has no time dependence.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_DYNAMIC** = 2
The problem is dynamic.
- INTEGER(INTG), parameter **PROBLEM_ROUTINES::PROBLEM_QUASISTATIC** = 3
The problem is quasi-static.
- TYPE(PROBLEMS_TYPE), target **PROBLEM_ROUTINES::PROBLEMS**

8.48.1 Detailed Description

Id

[problem_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all problem routines.

8.48.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by

deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [problem_routines.f90](#).

8.49 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/region_routines.f90 File Reference

Id

[region_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET](#)
- interface [REGION_ROUTINES::REGION_LABEL_SET](#)

Namespaces

- namespace [REGION_ROUTINES](#)

This module contains all region routines.

Functions

- TYPE(COORDINATE_SYSTEM_TYPE) [REGION_ROUTINES::REGION_COORDINATE_SYSTEM_GET](#) (REGION, ERR, ERROR)
- subroutine [REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_NUMBER](#) (USER_NUMBER, COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET_PTR](#) (REGION, COORDINATE_SYSTEM, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_CREATE_FINISH](#) (REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_CREATE_START](#) (USER_NUMBER, REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_DESTROY](#) (USER_NUMBER, ERR, ERROR,*)
- TYPE(VARYING_STRING) [REGION_ROUTINES::REGION_LABEL_GET](#) (REGION, ERR, ERROR)
- subroutine [REGION_ROUTINES::REGION_LABEL_SET_NUMBER](#) (USER_NUMBER, LABEL, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_LABEL_SET_PTR](#) (REGION, LABEL, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_SUB_REGION_CREATE_START](#) (USER_NUMBER, PARENT_REGION, SUB_REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_SUB_REGION_CREATE_FINISH](#) (REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_USER_NUMBER_FIND](#) (USER_NUMBER, REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGION_USER_NUMBER_FIND_PTR](#) (USER_NUMBER, REGION, START_REGION, ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGIONS_INITIALISE](#) (ERR, ERROR,*)
- subroutine [REGION_ROUTINES::REGIONS_FINALISE](#) (ERR, ERROR,*)

Variables

- TYPE(REGION_TYPE), target [REGION_ROUTINES::GLOBAL_REGION](#)

8.49.1 Detailed Description

Id

[region_routines.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all region routines.

8.49.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [region_routines.f90](#).

8.50 **d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solution_- mapping_routines.f90 File Reference**

8.51 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_- matrices_routines.f90 File Reference

Id

[solver_matrices_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Namespaces

- namespace [SOLVER_MATRICES_ROUTINES](#)
This module handles all solver matrix and rhs routines.

Functions

- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_FINISH](#) (SOLVER_MATRICES, ERR, ERROR,*)
Finishes the process of creating the solver matrices.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_CREATE_START](#) (SOLVER, SOLVER_MATRICES, ERR, ERROR,*)
Starts the process of creating the solver matrices.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_DESTROY](#) (SOLVER_MATRICES, ERR, ERROR,*)
Destroys the solver matrices.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_FINALISE](#) (SOLVER_MATRICES, ERR, ERROR,*)
Finalises the solver matrices and deallocates all memory.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_INITIALISE](#) (SOLVER, ERR, ERROR,*)
Initialises the solver matrices for a solver.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_LIBRARY_TYPE_SET](#) (SOLVER_MATRICES, LIBRARY_TYPE, ERR, ERROR,*)
Sets the library type for the solver matrices (and vectors).
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_OUTPUT](#) (ID, SOLVER_MATRICES, ERR, ERROR,*)
Outputs the solver matrices.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRICES_STORAGE_TYPE_SET](#) (SOLVER_MATRICES, STORAGE_TYPE, ERR, ERROR,*)
Sets the storage type (sparsity) of the solver matrices.
- subroutine [SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_STRUCTURE_-CALCULATE](#) (SOLVER_MATRIX, NUMBER_OF_NON_ZEROS, ROW_INDICES, COLUMN_INDICES, ERR, ERROR,*)

Calculates the structure (sparsity) of the solver matrix from the soluton mapping.

- subroutine **SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_FINALISE** (SOLVER_-
MATRIX, ERR, ERROR,*)

Finalises a solver matrix and deallocates all memory.

- subroutine **SOLVER_MATRICES_ROUTINES::SOLVER_MATRIX_INITIALISE** (SOLVER_-
MATRICES, MATRIX_NUMBER, ERR, ERROR,*)

Initialises a solver matrix.

8.51.1 Detailed Description

Id

[solver_matrices_routines.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module handles all solver matrix and rhs routines.

8.51.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [solver_matrices_routines.f90](#).

8.52 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_routines.f90 File Reference

Id

[solver_routines.f90](#) 20 2007-05-28 20:22:52Z cpb

Namespaces

- namespace [SOLVER_ROUTINES](#)

This module handles all solver routines.

Functions

- subroutine [SOLVER_ROUTINES::SOLVER_CREATE_FINISH](#) (SOLVER, ERR, ERROR,*)

Finishes the process of creating a solver for a problem solution.
- subroutine [SOLVER_ROUTINES::SOLVER_CREATE_START](#) (SOLUTION, SOLVE_TYPE, SOLVER, ERR, ERROR,*)

Starts the process of creating a solver for a problem solution.
- subroutine [SOLVER_ROUTINES::SOLVER_DESTROY](#) (SOLVER, ERR, ERROR,*)

Destroy a problem solver.
- subroutine [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_CREATE_FINISH](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Finishes the process of creating a eigenproblem solver.
- subroutine [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_FINALISE](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Finalise a eigenproblem solver for a problem solver.
- subroutine [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_INITIALISE](#) (SOLVER, ERR, ERROR,*)

Initialise a eigenproblem solver for a problem solver.
- subroutine [SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_SOLVE](#) (EIGENPROBLEM_SOLVER, ERR, ERROR,*)

Solve a eigenproblem solver.
- subroutine [SOLVER_ROUTINES::SOLVER_FINALISE](#) (SOLVER, ERR, ERROR,*)

Finalises a problem solver and deallocates all memory.
- subroutine [SOLVER_ROUTINES::SOLVER_INITIALISE](#) (SOLUTION, SOLVE_TYPE, ERR, ERROR,*)

Initialise a solver for a problem solution.
- subroutine [SOLVER_ROUTINES::SOLVER_LIBRARY_SET](#) (SOLVER, SOLVER_LIBRARY, ERR, ERROR,*)

Sets/changes the type of library to use for the solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_CREATE_FINISH** (LINEAR_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_CREATE_FINISH** (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear direct solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_FINALISE** (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Finalise a direct linear solver for a linear solver and deallocate all memory.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_INITIALISE** (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Initialise a direct linear solver for a linear solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE** (LINEAR_DIRECT_SOLVER, ERR, ERROR,*)

Solve a linear direct solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_TYPE_SET** (SOLVER, DIRECT_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of direct linear solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_FINALISE** (LINEAR_SOLVER, ERR, ERROR,*)

Finalise a linear solver for a problem solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_INITIALISE** (SOLVER, ERR, ERROR,*)

Initialise a linear solver for a problem solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_ABSOLUTE_TOLERANCE_SET** (SOLVER, ABSOLUTE_TOLERANCE, ERR, ERROR,*)

Sets/changes the maximum absolute tolerance for an iterative linear solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_CREATE_FINISH** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Finishes the process of creating a linear iterative solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_DIVERGENCE_TOLERANCE_SET** (SOLVER, DIVERGENCE_TOLERANCE, ERR, ERROR,*)

Sets/changes the maximum divergence tolerance for an iterative linear solver.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_FINALISE** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Finalise an iterative linear solver for a linear solver and deallocate all memory.

- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_INITIALISE** (LINEAR_SOLVER, ERR, ERROR,*)

Initialise an iterative linear solver for a linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_MAXIMUM_ITERATIONS_SET** (SOLVER, MAXIMUM_ITERATIONS, ERR, ERROR,*)

Sets/changes the maximum number of iterations for an iterative linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_PRECONDITIONER_TYPE_SET** (SOLVER, ITERATIVE_PRECONDITIONER_TYPE, ERR, ERROR,*)

Sets/changes the type of preconditioner for an iterative linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_RELATIVE_TOLERANCE_SET** (SOLVER, RELATIVE_TOLERANCE, ERR, ERROR,*)

Sets/changes the relative tolerance for an iterative linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE** (LINEAR_ITERATIVE_SOLVER, ERR, ERROR,*)

Solves a linear iterative linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_TYPE_SET** (SOLVER, ITERATIVE_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of iterative linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_SOLVE** (LINEAR_SOLVER, ERR, ERROR,*)

Solve a linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_LINEAR_TYPE_SET** (SOLVER, LINEAR_SOLVER_TYPE, ERR, ERROR,*)

Sets/changes the type of linear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_MATRICES_ASSEMBLE** (SOLVER, ERR, ERROR,*)

Assembles the solver matrices and rhs from the equations.
- subroutine **SOLVER_ROUTINES::SOLVER_NONLINEAR_CREATE_FINISH** (NONLINEAR_SOLVER, ERR, ERROR,*)

Finishes the process of creating a nonlinear solver.
- subroutine **SOLVER_ROUTINES::SOLVER_NONLINEAR_FINALISE** (NONLINEAR_SOLVER, ERR, ERROR,*)

Finalise a nonlinear solver for a problem solver.
- subroutine **SOLVER_ROUTINES::SOLVER_NONLINEAR_INITIALISE** (SOLVER, ERR, ERROR,*)

Initialise a nonlinear solver for a problem solver.
- subroutine **SOLVER_ROUTINES::SOLVER_NONLINEAR_SOLVE** (NONLINEAR_SOLVER, ERR, ERROR,*)

Solve a nonlinear solver for a problem solver.

- subroutine `SOLVER_ROUTINES::SOLVER_OUTPUT_TYPE_SET` (SOLVER, OUTPUT_TYPE, ERR, ERROR,*)

Sets/changes the output type for a solver.
- subroutine `SOLVER_ROUTINES::SOLVER_SPARSITY_TYPE_SET` (SOLVER, SPARSITY_TYPE, ERR, ERROR,*)

Sets/changes the sparsity type for a solver.
- subroutine `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_CREATE_FINISH` (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Finishes the process of creating a time integration solver.
- subroutine `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_FINALISE` (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Finalise a time integration solver for a problem solver.
- subroutine `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_INITIALISE` (SOLVER, ERR, ERROR,*)

Initialise a time integration solver for a problem solver.
- subroutine `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_SOLVE` (TIME_INTEGRATION_SOLVER, ERR, ERROR,*)

Solve a time integration solver.
- subroutine `SOLVER_ROUTINES::SOLVER_SOLVE` (SOLVER, ERR, ERROR,*)

Solve the problem.
- subroutine `SOLVER_ROUTINES::SOLVER_VARIABLES_UPDATE` (SOLVER, ERR, ERROR,*)

Updates the dependent variables from the solver solution.

Variables

- INTEGER(INTG), parameter `SOLVER_ROUTINES::SOLVER_LINEAR_TYPE = 1`

Linear solution solver.
- INTEGER(INTG), parameter `SOLVER_ROUTINES::SOLVER_NONLINEAR_TYPE = 2`

A nonlinear solution solver.
- INTEGER(INTG), parameter `SOLVER_ROUTINES::SOLVER_TIME_INTEGRATION_TYPE = 3`

A time integration solver.
- INTEGER(INTG), parameter `SOLVER_ROUTINES::SOLVER_EIGENPROBLEM_TYPE = 4`

A eigenproblem type.
- INTEGER(INTG), parameter `SOLVER_ROUTINES::SOLVER_CMISS_LIBRARY = LIBRARY_CMISS_TYPE`

CMISS (internal) solver library.

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_PETSC_LIBRARY = LIBRARY_-_PETSC_TYPE**
PETSc solver library.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_LINEAR_DIRECT_SOLVE_-_TYPE = 1**
Direct linear solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_LINEAR_ITERATIVE_SOLVE_-_TYPE = 2**
Iterative linear solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_DIRECT_LU = 1**
LU direct linear solver.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_DIRECT_CHOLESKY = 2**
Cholesky direct linear solver.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_DIRECT_SVD = 3**
SVD direct linear solver.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_RICHARDSON = 1**
Richardson iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CHEBYCHEV = 2**
Chebychev iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJUGATE_-_GRADIENT = 3**
Conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICONJUGATE_-_GRADIENT = 4**
Bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_GMRES = 5**
Generalised minimum residual iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BICGSTAB = 6**
Stabalised bi-conjugate gradient iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_CONJGRAD_-_SQUARED = 7**
Conjugate gradient squared iterative solver type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_NO_-_PRECONDITIONER = 0**
No preconditioner type.

- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_JACOBI_PRECONDITIONER** = 1
Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_BLOCK_JACOBI_PRECONDITIONER** = 2
Iterative block Jacobi preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_SOR_PRECONDITIONER** = 3
Successive over relaxation preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_CHOLESKY_PRECONDITIONER** = 4
Incomplete Cholesky preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_INCOMPLETE_LU_PRECONDITIONER** = 5
Incomplete LU preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ITERATIVE_ADDITIVE_SCHWARZ_PRECONDITIONER** = 6
Additive Schwarz preconditioner type.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_NO_OUTPUT** = 0
No output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_TIMING_OUTPUT** = 1
Timing output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_SOLVER_OUTPUT** = 2
Timing and solver specific output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_MATRIX_OUTPUT** = 3
Timing and solver specific output and solution matrices output from the solver routines.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_SPARSE_MATRICES** = 1
Use sparse solver matrices.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_FULL_MATRICES** = 2
Use fully populated solver matrices.
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_EULER_INTEGRATOR** = 1
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_IMPROVED_EULER_INTEGRATOR** = 2
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_4TH_RUNGE_KUTTA_INTEGRATOR** = 3
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_ADAMS_MOULTON_INTEGRATOR** = 4
- INTEGER(INTG), parameter **SOLVER_ROUTINES::SOLVER_LSODA_INTEGRATOR** = 5

8.52.1 Detailed Description

Id

[solver_routines.f90](#) 20 2007-05-28 20:22:52Z cpb

Author:

Chris Bradley This module handles all solver routines.

8.52.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [solver_routines.f90](#).

8.53 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/sorting.f90

File Reference

Id

[sorting.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [SORTING::BUBBLE_SORT](#)
- interface [SORTING::HEAP_SORT](#)
- interface [SORTING::SHELL_SORT](#)

Namespaces

- namespace [SORTING](#)

This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

Functions

- subroutine [SORTING::BUBBLE_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [SORTING::BUBBLE_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [SORTING::BUBBLE_SORT_DP](#) (A, ERR, ERROR,*)
- subroutine [SORTING::HEAP_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [SORTING::HEAP_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [SORTING::HEAP_SORT_DP](#) (A, ERR, ERROR,*)
- subroutine [SORTING::SHELL_SORT_INTG](#) (A, ERR, ERROR,*)
- subroutine [SORTING::SHELL_SORT_SP](#) (A, ERR, ERROR,*)
- subroutine [SORTING::SHELL_SORT_DP](#) (A, ERR, ERROR,*)

8.53.1 Detailed Description

Id

[sorting.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all procedures for sorting. NOTE: THE ROUTINES IN THIS MODULE HAVE NOT BEEN TESTED!!!

8.53.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [sorting.f90](#).

8.54 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/strings.f90

File Reference

Id

[strings.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [STRINGS::IS_ABBREVIATION](#)

Returns .TRUE. if a supplied string is a valid abbreviation of a second supplied string.
- interface [STRINGS::LIST_TO_CHARACTER](#)

Converts a list to its equivalent character string representation.
- interface [STRINGS::NUMBER_TO_CHARACTER](#)

Converts a number to its equivalent character string representation.
- interface [STRINGS::NUMBER_TO_VSTRING](#)

Converts a number to its equivalent varying string representation.
- interface [STRINGS::STRING_TO_DOUBLE](#)

Converts a string representation of a number to a double precision number.
- interface [STRINGS::STRING_TO_INTEGER](#)

Converts a string representation of a number to an integer.
- interface [STRINGS::STRING_TO_LONG_INTEGER](#)

Converts a string representation of a number to a long integer.
- interface [STRINGS::STRING_TO_LOGICAL](#)

Converts a string representation of a boolean value (TRUE or FALSE) to a logical.
- interface [STRINGS::STRING_TO_SINGLE](#)

Converts a string representation of a number to a single precision number.
- interface [STRINGS::CHARACTER_TO_LOWERCASE](#)

Returns a character string which is the lowercase equivalent of the supplied string.
- interface [STRINGS::VSTRING_TO_LOWERCASE](#)

Returns a varying string which is the lowercase equivalent of the supplied string.
- interface [STRINGS::CHARACTER_TO_UPPERCASE](#)

Returns a character string which is the uppercase equivalent of the supplied string.
- interface [STRINGS::VSTRING_TO_UPPERCASE](#)

Returns a varying string which is the uppercase equivalent of the supplied string.

Namespaces

- namespace **STRINGS**

This module contains all string manipulation and transformation routines.

Functions

- LOGICAL **STRINGS::IS_ABBREVIATION_C_C** (SHORT, LONG, MIN_NUM_-CHARACTERS)

IS_ABBREVIATION returns .TRUE. if the character string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

- LOGICAL **STRINGS::IS_ABBREVIATION_C_VS** (SHORT, LONG, MIN_NUM_-CHARACTERS)

IS_ABBREVIATION returns .TRUE. if the character string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

- LOGICAL **STRINGS::IS_ABBREVIATION_VS_C** (SHORT, LONG, MIN_NUM_-CHARACTERS)

IS_ABBREVIATION returns .TRUE. if the varying string SHORT is an abbreviation of the character string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

- LOGICAL **STRINGS::IS_ABBREVIATION_VS_VS** (SHORT, LONG, MIN_NUM_-CHARACTERS)

IS_ABBREVIATION returns .TRUE. if the varying string SHORT is an abbreviation of the varying string LONG. SHORT must be at least MIN_NUM_CHARACTERS long.

- LOGICAL **STRINGS::IS_DIGIT** (CHARAC)

IS_DIGIT returns .TRUE. if the character CHARAC is a digit character (i.e. 0..9).

- LOGICAL **STRINGS::IS_LETTER** (CHARAC)

IS_LETTER returns .TRUE. if the character CHARAC is a letter character (i.e. A..Z or a..z).

- LOGICAL **STRINGS::IS_LOWERCASE** (CHARC)

Returns .TRUE. if the supplied character is a lowercase character.

- LOGICAL **STRINGS::IS_UPPERCASE** (CHARC)

Returns .TRUE. if the supplied character is an uppercase character.

- LOGICAL **STRINGS::IS_WHITESPACE** (CHARAC)

IS_WHITESPACE returns .TRUE. if the character CHARAC is a whitespace character (i.e. space, tabs, etc.).

- CHARACTER(LEN=MAXSTRLEN) **STRINGS::LIST_TO_CHARACTER_C** (NUMBER_IN_-LIST, LIST, FORMAT, ERR, ERROR, LIST_LENGTHS)

Converts a character list to its equivalent character string representation as determined by the supplied format. If present, the optional argument LIST_LENGTHS is used for the lengths of each list elements length otherwise the trimmed length is used. NOTE: The FORMAT is ignored for this child FUNCTION.

- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST_TO_CHARACTER_INTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts an integer list to its equivalent character string representation as determined by the supplied format.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST_TO_CHARACTER_LINTG](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts an long integer list to its equivalent character string representation as determined by the supplied format.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST_TO_CHARACTER_L](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a logical list to its equivalent character string representation as determined by the supplied format string. The FORMAT is ignored for this child FUNCTION.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST_TO_CHARACTER_SP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a single precision list to its equivalent character string representation as determined by the supplied format string.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LIST_TO_CHARACTER_DP](#) (NUMBER_IN_LIST, LIST, FORMAT, ERR, ERROR)

Converts a double precision list to its equivalent character string representation as determined by the supplied format string.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::LOGICAL_TO_CHARACTER](#) (LOGICAL_VALUE, ERR, ERROR)

Converts a logical value to either a "TRUE" or "FALSE" character string.
- TYPE(VARYING_STRING) [STRINGS::LOGICAL_TO_VSTRING](#) (LOGICALVALUE, ERR, ERROR)

Converts a logical value to either a "TRUE" or "FALSE" varying string.
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER_TO_CHARACTER_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts an integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER_TO_CHARACTER_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts a long integer number to its equivalent character string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER_TO_CHARACTER_SP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent character string representation as determined by the supplied format string. NOTE: If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*
- CHARACTER(LEN=MAXSTRLEN) [STRINGS::NUMBER_TO_CHARACTER_DP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent character string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE(VARYING_STRING) [STRINGS::NUMBER_TO_VSTRING_INTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts an integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g. "I3".

- TYPE(VARYING_STRING) [STRINGS::NUMBER_TO_VSTRING_LINTG](#) (NUMBER, FORMAT, ERR, ERROR)

Converts a long integer number to its equivalent varying string representation as determined by the supplied format. The format is of the form of a standard Fortran integer format e.g., "I3".

- TYPE(VARYING_STRING) [STRINGS::NUMBER_TO_VSTRING_SP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a single precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- TYPE(VARYING_STRING) [STRINGS::NUMBER_TO_VSTRING_DP](#) (NUMBER, FORMAT, ERR, ERROR)

*Converts a double precision number to its equivalent varying string representation as determined by the supplied format string. Note If FORMAT is an asterisk followed by a number between 1 and 32 the format will be chosen to maximise the number of significant digits, e.g., FORMAT="*8" will return a string of 8 characters representing the supplied number in either F8.? or E8.? format depending on its magnitude.*

- REAL(DP) [STRINGS::STRING_TO_DOUBLE_C](#) (STRING, ERR, ERROR)

Converts a character string representation of a number to a double precision number.

- REAL(DP) [STRINGS::STRING_TO_DOUBLE_VS](#) (STRING, ERR, ERROR)

Converts a varying string representation of a number to a double precision number.

- INTEGER(INTG) [STRINGS::STRING_TO_INTEGER_C](#) (STRING, ERR, ERROR)

Converts a character string representation of a number to an integer.

- INTEGER(INTG) [STRINGS::STRING_TO_INTEGER_VS](#) (STRING, ERR, ERROR)

Converts a varying string representation of a number to an integer.

- INTEGER(LINTG) [STRINGS::STRING_TO_LONG_INTEGER_C](#) (STRING, ERR, ERROR)

Converts a character string representation of a number to a long integer.

- INTEGER(LINTG) [STRINGS::STRING_TO_LONG_INTEGER_VS](#) (STRING, ERR, ERROR)

Converts a varying string representation of a number to a long integer.

- LOGICAL [STRINGS::STRING_TO_LOGICAL_C](#) (STRING, ERR, ERROR)

Converts a character string representation of a boolean (TRUE or FALSE) to a logical.

- LOGICAL [STRINGS::STRING_TO_LOGICAL_VS](#) (STRING, ERR, ERROR)

Converts a varying string representation of a boolean (TRUE or FALSE) to a logical.

- REAL(SP) **STRINGS::STRING_TO_SINGLE_C** (STRING, ERR, ERROR)
Converts a character string representation of a number to a single precision number.
- REAL(SP) **STRINGS::STRING_TO_SINGLE_VS** (STRING, ERR, ERROR)
Converts a varying string representation of a number to a single precision number.
- function **STRINGS::CHARACTER_TO_LOWERCASE_C** (STRING)
• function **STRINGS::CHARACTER_TO_LOWERCASE_VS** (STRING)
Returns a character string that is the lowercase equivalent of the supplied varying string.
- TYPE(VARYING_STRING) **STRINGS::VSTRING_TO_LOWERCASE_C** (STRING)
Returns a varying string that is the lowercase equivalent of the supplied character string.
- TYPE(VARYING_STRING) **STRINGS::VSTRING_TO_LOWERCASE_VS** (STRING)
Returns a varying string that is the lowercase equivalent of the supplied varying string.
- function **STRINGS::CHARACTER_TO_UPPERCASE_C** (STRING)
Returns a character string which is uppercase equivalent of the supplied character string.
- function **STRINGS::CHARACTER_TO_UPPERCASE_VS** (STRING)
Returns a character string which is uppercase equivalent of the supplied varying string.
- TYPE(VARYING_STRING) **STRINGS::VSTRING_TO_UPPERCASE_C** (STRING)
Returns a varying string which is uppercase equivalent of the supplied character string.
- TYPE(VARYING_STRING) **STRINGS::VSTRING_TO_UPPERCASE_VS** (STRING)
Returns a varying string which is uppercase equivalent of the supplied varying string.

8.54.1 Detailed Description

Id

[strings.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains all string manipulation and transformation routines.

8.54.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [strings.f90](#).

8.55 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_c.c File Reference

Include dependency graph for timer_c.c:

Defines

- #define **USER_CPU** 1
- #define **SYSTEM_CPU** 2
- #define **TOTAL_CPU** 3

Functions

- void **CPUTimer** (double *return_time, int *flag, int *err, char *error_string)

8.55.1 Define Documentation

8.55.1.1 #define SYSTEM_CPU 2

Referenced by CPUTimer().

8.55.1.2 #define TOTAL_CPU 3

Referenced by CPUTimer().

8.55.1.3 #define USER_CPU 1

Referenced by CPUTimer().

8.55.2 Function Documentation

8.55.2.1 void **CPUTimer** (*double *return_time, int *flag, int *err, char *error_string*)

Definition at line 107 of file timer_c.c.

References SYSTEM_CPU, TOTAL_CPU, and USER_CPU.

8.56 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_f.f90 File Reference

Id

[timer_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Classes

- interface [TIMER::interface](#)

Namespaces

- namespace [TIMER](#)

This module contains routines for timing the program.

Functions

- subroutine [TIMER::CPU_TIMER](#) (TIME_TYPE, TIME, ERR, ERROR,*)

Variables

- INTEGER(INTG), parameter [TIMER::USER_CPU](#) = 1
- INTEGER(INTG), parameter [TIMER::SYSTEM_CPU](#) = 2
- INTEGER(INTG), parameter [TIMER::TOTAL_CPU](#) = 3

8.56.1 Detailed Description

Id

[timer_f.f90](#) 27 2007-07-24 16:52:51Z cpb

Author:

Chris Bradley This module contains routines for timing the program.

8.56.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University

of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [timer_f.f90](#).

8.57 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/trees.f90 File Reference

Id

[trees.f90](#) 2 2007-07-27 08:35:14Z cpb

Classes

- struct [TREES::TREE_NODE_TYPE](#)
- struct [TREES::TREE_TYPE](#)

Namespaces

- namespace [TREES](#)

Implements trees of base types.

Functions

- subroutine [TREES::TREE_CREATE_FINISH](#) (TREE, ERR, ERROR,*)
Finishes the creation of a tree created with TREE_CREATE_START.
- subroutine [TREES::TREE_CREATE_START](#) (TREE, ERR, ERROR,*)
Starts the creation of a tree and returns a pointer to the created tree.
- subroutine [TREES::TREE_DESTROY](#) (TREE, ERR, ERROR,*)
Destroys a tree.
- subroutine [TREES::TREE_DETACH_AND_DESTROY](#) (TREE, NUMBER_IN_TREE, TREE_VALUES, ERR, ERROR,*)
Detaches the tree values and returns them as a pointer to the an array and then destroys the tree.
- subroutine [TREES::TREE_DETACH_IN_ORDER](#) (TREE, X, COUNT, TREE_VALUES, ERR, ERROR,*)
Detaches the tree values in order from the specified tree node and adds them to the tree values array.
- subroutine [TREES::TREE_FINALISE](#) (TREE, ERR, ERROR,*)
Finalises a tree and deallocates all memory.
- subroutine [TREES::TREE_INITIALISE](#) (TREE, ERR, ERROR,*)
Initialises a tree.
- subroutine [TREES::TREE_INSERT_TYPE_SET](#) (TREE, INSERT_TYPE, ERR, ERROR,*)
Sets/changes the insert type for a tree.
- subroutine [TREES::TREE_ITEM_DELETE](#) (TREE, KEY, ERR, ERROR,*)
Deletes a tree node specified by a key from a tree.

- subroutine [TREES::TREE_ITEM_INSERT](#) (TREE, KEY, VALUE, INSERT_STATUS, ERR, ERROR,*)

Inserts a tree node into a red-black tree.
- subroutine [TREES::TREE_NODE_FINALISE](#) (TREE, TREE_NODE, ERR, ERROR,*)

Finalises a tree node and deallocates all memory.
- subroutine [TREES::TREE_NODE_INITIALISE](#) (TREE, TREE_NODE, ERR, ERROR,*)

Initialises a tree node.
- subroutine [TREES::TREE_NODE_KEY_GET](#) (TREE, TREE_NODE, KEY, ERR, ERROR,*)

Gets the key at a specified tree node.
- subroutine [TREES::TREE_NODE_VALUE_GET](#) (TREE, TREE_NODE, VALUE, ERR, ERROR,*)

Gets the value at a specified tree node.
- subroutine [TREES::TREE_NODE_VALUE_SET](#) (TREE, TREE_NODE, VALUE, ERR, ERROR,*)

Sets the value at a specified tree node.
- subroutine [TREES::TREE_OUTPUT](#) (ID, TREE, ERR, ERROR,*)

Outputs a tree to the specified output stream ID.
- subroutine [TREES::TREE_OUTPUT_IN_ORDER](#) (ID, TREE, X, ERR, ERROR,*)

Outputs a tree in order to the specified output stream ID from the specified tree node.
- TYPE(TREE_NODE_TYPE), pointer [TREES::TREE_PREDECESSOR](#) (TREE, X, ERR, ERROR)

Returns the predecessor of a tree at a specified tree node.
- subroutine [TREES::TREE_SEARCH](#) (TREE, KEY, X, ERR, ERROR,*)

Searches a tree to see if it contains a key.
- TYPE(TREE_NODE_TYPE), pointer [TREES::TREE_SUCCESSOR](#) (TREE, X, ERR, ERROR)

Returns the successor of a tree at a specified tree node.

Variables

- INTEGER(INTG), parameter [TREES::TREE_BLACK_NODE](#) = 0

The black colour type for a tree node.
- INTEGER(INTG), parameter [TREES::TREE_RED_NODE](#) = 1

The red colour type for a tree node.
- INTEGER(INTG), parameter [TREES::TREE_NODE_INSERT_SUCESSFUL](#) = 1

Successful insert status.
- INTEGER(INTG), parameter [TREES::TREE_NODE_DUPLICATE_KEY](#) = 2

Duplicate key found for those trees that do not allow duplicate keys.

- INTEGER(INTG), parameter [TREES::TREE_DUPLICATES_ALLOWED_TYPE](#) = 1
Duplicate keys allowed tree type.
- INTEGER(INTG), parameter [TREES::TREE_NO_DUPLICATES_ALLOWED](#) = 2
No duplicate keys allowed tree type.

8.57.1 Detailed Description

Id

[trees.f90](#) 2 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley Implements trees of base types.

8.57.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [trees.f90](#).

8.58 d:/Users/tyu011/workspace/OpenCMISS-trunk/src/types.f90 File Reference

Id

[types.f90](#) 28 2007-07-27 08:35:14Z cpb

Classes

- struct [TYPES::QUADRATURE_SCHEME_TYPE](#)
Contains information for a particular quadrature scheme.
- struct [TYPES::QUADRATURE_SCHEME_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [QUADRATURE_SCHEME_TYPE](#).
- struct [TYPES::QUADRATURE_TYPE](#)
Contains information on the quadrature to be used for integrating a basis.
- struct [TYPES::BASIS_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [BASIS_TYPE](#).
- struct [TYPES::BASIS_TYPE](#)
Contains all information about a basis .
- struct [TYPES::BASIS_FUNCTIONS_TYPE](#)
Contains information on the defined basis functions.
- struct [TYPES::COORDINATE_SYSTEM_TYPE](#)
Contains information on a coordinate system.
- struct [TYPES::NODE_TYPE](#)
Contains information about a node.
- struct [TYPES::NODES_TYPE](#)
Contains information on the nodes defined on a region.
- struct [TYPES::MESH_DOFS_TYPE](#)
Contains information on the dofs for a mesh.
- struct [TYPES::MESH_ELEMENT_TYPE](#)
Contains the information for an element in a mesh.
- struct [TYPES::MESH_ELEMENTS_TYPE](#)
Contains the information for the elements of a mesh.
- struct [TYPES::MESH_NODE_TYPE](#)
Contains the topology information for a global node of a mesh.
- struct [TYPES::MESH_NODES_TYPE](#)
Contains the information for the nodes of a mesh.

- struct **TYPES::MESH_TOPOLOGY_TYPE**
Contains information on the (global) topology of a mesh.
- struct **TYPES::MESH_TOPOLOGY_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **MESH_TOPOLOGY_TYPE**.*
- struct **TYPES::MESH_TYPE**
Contains information on a mesh defined on a region.
- struct **TYPES::MESH_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **MESH_TYPE**.*
- struct **TYPES::MESHES_TYPE**
Contains information on the meshes defined on a region.
- struct **TYPES::GENERATED_MESH_REGULAR_TYPE**
Contains information on a generated regular mesh.
- struct **TYPES::GENERATED_MESH_TYPE**
Contains information on a generated mesh.
- struct **TYPES::DOMAIN_DOFS_TYPE**
Contains information on the degrees-of-freedom (dofs) for a domain.
- struct **TYPES::DOMAIN_LINE_TYPE**
Contains the information for a line in a domain.
- struct **TYPES::DOMAIN_LINE_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **DOMAIN_LINE_TYPE**.*
- struct **TYPES::DOMAIN_LINES_TYPE**
Contains the topology information for the lines of a domain.
- struct **TYPES::DOMAIN_FACE_TYPE**
Contains the information for a face in a domain.
- struct **TYPES::DOMAIN_FACE_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **FIELD_VARIABLE_TYPE**.*
- struct **TYPES::DOMAIN_FACES_TYPE**
Contains the topology information for the faces of a domain.
- struct **TYPES::DOMAIN_ELEMENT_TYPE**
Contains the information for an element in a domain.
- struct **TYPES::DOMAIN_ELEMENTS_TYPE**
Contains the topology information for the elements of a domain.
- struct **TYPES::DOMAIN_NODE_TYPE**

Contains the topology information for a local node of a domain.

- struct [TYPES::DOMAIN_NODES_TYPE](#)

Contains the topology information for the nodes of a domain.

- struct [TYPES::DOMAIN_TOPOLOGY_TYPE](#)

Contains the topology information for a domain.

- struct [TYPES::DISTRIBUTED_VECTOR_TRANSFER_TYPE](#)

Contains the information for an adjacent domain for transferring the ghost data of a distributed vector to/from the current domain.

- struct [TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE](#)

Contains information for a [CMISS](#) distributed vector.

- struct [TYPES::DISTRIBUTED_VECTOR_PETSC_TYPE](#)

Contains information for a PETSc distributed vector.

- struct [TYPES::DISTRIBUTED_VECTOR_TYPE](#)

Contains the information for a vector that is distributed across a number of domains.

- struct [TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE](#)

Contains information for a [CMISS](#) distributed matrix.

- struct [TYPES::DISTRIBUTED_MATRIX_PETSC_TYPE](#)

Contains information for a PETSc distributed matrix.

- struct [TYPES::DISTRIBUTED_MATRIX_TYPE](#)

Contains the information for a matrix that is distributed across a number of domains.

- struct [TYPES::VECTOR_TYPE](#)

Contains information for a vector.

- struct [TYPES::MATRIX_TYPE](#)

Contains information for a matrix.

- struct [TYPES::DOMAIN_ADJACENT_DOMAIN_TYPE](#)

Contains the information on an adjacent domain to a domain in a domain mapping.

- struct [TYPES::DOMAIN_GLOBAL_MAPPING_TYPE](#)

Contains the local information for a global mapping number for a domain mapping.

- struct [TYPES::DOMAIN_MAPPING_TYPE](#)

Contains information on the domain mappings (i.e., local and global numberings).

- struct [TYPES::DOMAIN_MAPPINGS_TYPE](#)

Contains information on the domain decomposition mappings.

- struct [TYPES::DOMAIN_TYPE](#)

A pointer to the domain decomposition for this domain.

- struct **TYPES::DOMAIN_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **DOMAIN_TYPE**.*
- struct **TYPES::DECOMPOSITION_LINE_TYPE**
Contains the information for a line in a decomposition.
- struct **TYPES::DECOMPOSITION_LINES_TYPE**
Contains the topology information for the lines of a decomposition.
- struct **TYPES::DECOMPOSITION_ELEMENT_TYPE**
Contains the information for an element in a decomposition.
- struct **TYPES::DECOMPOSITION_ELEMENTS_TYPE**
Contains the topology information for the elements of a decomposition.
- struct **TYPES::DECOMPOSITION_TOPOLOGY_TYPE**
Contains the topology information for a decomposition.
- struct **TYPES::DECOMPOSITION_TYPE**
Contains information on the domain decomposition.
- struct **TYPES::DECOMPOSITION_PTR_TYPE**
*A buffer type to allow for an array of pointers to a **DECOMPOSITION_TYPE**.*
- struct **TYPES::DECOMPOSITIONS_TYPE**
Contains information on the domain decompositions defined on a mesh.
- struct **TYPES::FIELD_INTERPOLATED_POINT_METRICS_TYPE**
Contains the interpolated point coordinate metrics. Old CMISS name GL, GU, RG.
- struct **TYPES::FIELD_INTERPOLATED_POINT_TYPE**
Contains the interpolated value (and the derivatives wrt xi) of a field at a point. Old CMISS name XG.
- struct **TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE**
Contains the parameters required to interpolate a field variable within an element. Old CMISS name XE.
- struct **TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE**
Contains the geometric parameters (lines, faces, volumes etc.) for a geometric field decomposition.
- struct **TYPES::FIELD_SCALING_TYPE**
A type to hold the scale factors for the appropriate mesh component of a field.
- struct **TYPES::FIELD_SCALINGS_TYPE**
A type to hold the field scalings for the field.
- struct **TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE**
A type to hold the mapping from field dof numbers to field parameters (nodes, elements, etc).
- struct **TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE**

A type to hold the mapping from field parameters (nodes, elements, etc) to field dof numbers for a particular field variable component.

- struct [TYPES::FIELD_VARIABLE_COMPONENT_TYPE](#)
Contains information for a component of a field variable.
- struct [TYPES::FIELD_VARIABLE_TYPE](#)
Contains information for a field variable defined on a field.
- struct [TYPES::FIELD_VARIABLE_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_VARIABLE_TYPE](#).
- struct [TYPES::FIELD_CREATE_VALUES_CACHE_TYPE](#)
A type to temporarily hold (cache) the user modifiable values which are used to create a field.
- struct [TYPES::FIELD_MAPPINGS_TYPE](#)
The type containing the mappings for the field.
- struct [TYPES::FIELD_PARAMETER_SET_TYPE](#)
A type to hold the parameter sets for a field.
- struct [TYPES::FIELD_PARAMETER_SET_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_PARAMETER_SET_TYPE](#).
- struct [TYPES::FIELD_PARAMETER_SETS_TYPE](#)
A type to store the parameter sets for a field.
- struct [TYPES::FIELD_TYPE](#)
Contains information for a field defined on a region.
- struct [TYPES::FIELD_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [FIELD_TYPE](#).
- struct [TYPES::FIELDS_TYPE](#)
Contains information on the fields defined on a region.
- struct [TYPES::ELEMENT_MATRIX_TYPE](#)
Contains information for an element matrix.
- struct [TYPES::ELEMENT_VECTOR_TYPE](#)
Contains information for an element vector.
- struct [TYPES::EQUATIONS_MATRIX_TYPE](#)
Contains information about an equations matrix.
- struct [TYPES::EQUATIONS_MATRIX_PTR_TYPE](#)
- struct [TYPES::EQUATIONS_MATRICES_TYPE](#)
Contains information on the equations matrices and rhs vector.
- struct [TYPES::VARIABLE_TO_EQUATIONS_COLUMN_MAP_TYPE](#)

Contains the information about the mapping of a variable DOF to an equations matrix column.

- struct [TYPES::VARIABLE_TO_EQUATIONS_MATRICES_MAP_TYPE](#)
Contains the mapping for a dependent variable type to the equations matrices.
 - struct [TYPES::SOURCE_EQUATIONS_MATRICES_MAP_TYPE](#)
 - struct [TYPES::EQUATIONS_MATRIX_TO_VARIABLE_MAP_TYPE](#)
 - struct [TYPES::EQUATIONS_ROW_TO_VARIABLE_MAP_TYPE](#)
 - struct [TYPES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE](#)
 - struct [TYPES::EQUATIONS_MAPPING_TYPE](#)
 - struct [TYPES::EQUATIONS_INTERPOLATION_TYPE](#)
Contains information on the interpolation for the equations.
- struct [TYPES::EQUATIONS_LINEAR_DATA_TYPE](#)
Contains information on any data required for a linear solution.
- struct [TYPES::EQUATIONS_NONLINEAR_DATA_TYPE](#)
Contains information on any data required for a non-linear solution.
- struct [TYPES::EQUATIONS_TIME_DATA_TYPE](#)
Contains information on any data required for a time-dependent equations.
- struct [TYPES::EQUATIONS_TYPE](#)
Contains information about the equations in an equations set.
- struct [TYPES::EQUATIONS_SET_GEOMETRY_TYPE](#)
Contains information on the geometry for an equations set.
- struct [TYPES::EQUATIONS_SET_MATERIALS_TYPE](#)
- struct [TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE](#)
Contains information on the fixed conditions for the problem.
- struct [TYPES::EQUATIONS_SET_DEPENDENT_TYPE](#)
Contains information on the dependent variables for the equations set.
- struct [TYPES::EQUATIONS_SET_SOURCE_TYPE](#)
Contains information on the source for the equations set.
- struct [TYPES::EQUATIONS_SET_ANALYTIC_TYPE](#)
Contains information on the analytic setup for the equations set.
- struct [TYPES::EQUATIONS_SET_TYPE](#)
Contains information on an equations set.
 - struct [TYPES::EQUATIONS_SET_PTR_TYPE](#)
 - struct [TYPES::EQUATIONS_SETS_TYPE](#)
 - struct [TYPES::SOLVER_MATRIX_TYPE](#)
Contains information on the solver matrix.
- struct [TYPES::SOLVER_MATRIX_PTR_TYPE](#)

- struct [TYPES::SOLVER_MATRICES_TYPE](#)
Contains information on the solver matrices and rhs vector.
- struct [TYPES::LINEAR_DIRECT_SOLVER_TYPE](#)
Contains information for a direct linear solver.
- struct [TYPES::LINEAR_ITERATIVE_SOLVER_TYPE](#)
Contains information for a direct linear solver.
- struct [TYPES::LINEAR_SOLVER_TYPE](#)
Contains information for a linear solver.
- struct [TYPES::NONLINEAR_SOLVER_TYPE](#)
Contains information for a nonlinear solver.
- struct [TYPES::TIME_INTEGRATION_SOLVER_TYPE](#)
Contains information for a time integration solver.
- struct [TYPES::EIGENPROBLEM_SOLVER_TYPE](#)
Contains information for a time integration solver.
- struct [TYPES::SOLVER_TYPE](#)
Contains information on the type of solver to be used.
- struct [TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE](#)
- struct [TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE](#)
- struct [TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE](#)
A buffer type to allow for an array of pointers to a [EQUATIONS_TO_SOLVER_MAPS_TYPE](#).
- struct [TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE](#)
Contains information on the mappings between field variable dofs in an equation set and the solver matrix columns (solver dofs).
- struct [TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE](#)
Contains information on the equations to solver matrix mappings when indexing by solver matrix number.
- struct [TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE](#)
Contains information on the equations to solver matrix mappings when indexing by equations matrix number.
- struct [TYPES::EQUATIONS_ROW_TO_SOLVER_ROWS_MAP_TYPE](#)
Contains information on the mapping from the equations rows in an equations set to the solver rows.
- struct [TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE](#)
Contains information on the mappings from an equations set to the solver matrices.
- struct [TYPES::SOLVER_COL_TO_EQUATIONS_MAP_TYPE](#)
Contains information about the mapping from a solver matrix column to equations matrices and variables.
- struct [TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE](#)

Contains information about mapping the solver column (solver dof) to the field variable dofs in the equations set.

- struct [TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE](#)

Contains information about the mappings from a solver matrix to the equations in an equations set.

- struct [TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE](#)

Contains information on the mappings from a solver matrix to equations sets.

- struct [TYPES::SOLVER_ROW_TO_EQUATIONS_SET_MAP_TYPE](#)

Contains information on the mappings from a solver row to the equations rows.

- struct [TYPES::SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE](#)

Contains information about the cached create values for a solution mapping.

- struct [TYPES::SOLUTION_MAPPING_TYPE](#)

Contains information on the solution mapping between the global equation sets and the solver matrices.

- struct [TYPES::SOLUTION_TYPE](#)

Contains information on the solution of a problem.

- struct [TYPES::SOLUTION_PTR_TYPE](#)

- struct [TYPES::PROBLEM_LINEAR_DATA_TYPE](#)

Contains information on any data required for a linear solution.

- struct [TYPES::PROBLEM_NONLINEAR_DATA_TYPE](#)

Contains information on any data required for a non-linear solution.

- struct [TYPES::PROBLEM_TIME_DATA_TYPE](#)

Contains information on any data required for a time-dependent solution.

- struct [TYPES::PROBLEM_CONTROL_TYPE](#)

- struct [TYPES::PROBLEM_TYPE](#)

Contains information for a problem.

- struct [TYPES::PROBLEM_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [PROBLEM_TYPE](#).

- struct [TYPES::PROBLEMS_TYPE](#)

Contains information on the problems defined on a region.

- struct [TYPES::REGION_PTR_TYPE](#)

A buffer type to allow for an array of pointers to a [REGION_TYPE](#).

- struct [TYPES::REGION_TYPE](#)

Contains information for a region.

Namespaces

- namespace **TYPES**

This module contains all type definitions in order to avoid cyclic module references.

8.58.1 Detailed Description

Id

[types.f90](#) 28 2007-07-27 08:35:14Z cpb

Author:

Chris Bradley This module contains all type definitions in order to avoid cyclic module references.

8.58.2 LICENSE

Version: MPL 1.1/GPL 2.0/LGPL 2.1

The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at <http://www.mozilla.org/MPL/>

Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

The Original Code is openCMISS

The Initial Developer of the Original Code is University of Auckland, Auckland, New Zealand and University of Oxford, Oxford, United Kingdom. Portions created by the University of Auckland and University of Oxford are Copyright (C) 2007 by the University of Auckland and the University of Oxford. All Rights Reserved.

Contributor(s):

Alternatively, the contents of this file may be used under the terms of either the GNU General Public License Version 2 or later (the "GPL"), or the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), in which case the provisions of the GPL or the LGPL are applicable instead of those above. If you wish to allow use of your version of this file only under the terms of either the GPL or the LGPL, and not to allow others to use your version of this file under the terms of the MPL, indicate your decision by deleting the provisions above and replace them with the notice and other provisions required by the GPL or the LGPL. If you do not delete the provisions above, a recipient may use your version of this file under the terms of any one of the MPL, the GPL or the LGPL.

Definition in file [types.f90](#).

Index

_DP
 KINDS_ComplexKinds, 82

_SP
 KINDS_ComplexKinds, 83

_SYSTEM_TYPE_STRING
 COORDINATE_ROUTINES, 260

ABSOLUTE_TOLERANCE
 TYPES::LINEAR_ITERATIVE_SOLVER_TYPE, 1125

ACCESS_TYPE
 BINARY_FILE::BINARY_FILE_INFO_TYPE, 698

additional_doc/Installation.dox, 1209

additional_doc/Test.dox, 1210

ADDITIVE_CONSTANT
 TYPES::SOLVER_COL_TO_VARIABLE_MAP_TYPE, 1186

ADDITIVE_CONSTANTS
 TYPES::VARIABLE_TO_SOLVER_COL_MAP_TYPE, 1204

ADJACENT_DOMAINS
 TYPES::DOMAIN_MAPPING_TYPE, 1012

ADJACENT_DOMAINS_LIST
 TYPES::DOMAIN_MAPPING_TYPE, 1012

ADJACENT_DOMAINS_PTR
 TYPES::DOMAIN_MAPPING_TYPE, 1012

ADJACENT_ELEMENTS
 TYPES::DECOMPOSITION_ELEMENT_TYPE, 961

 TYPES::MESH_ELEMENT_TYPE, 1134

ADJACENT_LINES
 TYPES::DECOMPOSITION_LINE_TYPE, 965

adjustl_
 ISO_VARYING_STRING, 439

 ISO_VARYING_STRING::assignment(=), 817

adjustr_
 ISO_VARYING_STRING, 439

 ISO_VARYING_STRING::adjustr, 816

ALL_DIAG_TYPE
 BASE_ROUTINES_DiagnosticTypes, 26

ALL_TIMING_TYPE
 BASE_ROUTINES_TimingTypes, 28

ANALYTIC
 TYPES::EQUATIONS_SET_TYPE, 1065

ANALYTIC_FINISHED
 TYPES::EQUATIONS_SET_ANALYTIC_TYPE, 1054

AREAS
 TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE, 1086

BASE_ROUTINES, 133

 BASE_ROUTINES_FINALISE, 138

 BASE_ROUTINES_INITIALISE, 138

DIAG_ALL_SUBROUTINES, 171

DIAG_FILE_OPEN, 171

DIAG_FROM_SUBROUTINE, 171

DIAG_OR_TIMING, 171

DIAG_ROUTINE_LIST, 171

DIAGNOSTICS, 171

DIAGNOSTICS1, 171

DIAGNOSTICS2, 172

DIAGNOSTICS3, 172

DIAGNOSTICS4, 172

DIAGNOSTICS5, 172

DIAGNOSTICS_LEVEL1, 172

DIAGNOSTICS_LEVEL2, 172

DIAGNOSTICS_LEVEL3, 172

DIAGNOSTICS_LEVEL4, 173

DIAGNOSTICS_LEVEL5, 173

DIAGNOSTICS_SET_OFF, 138

DIAGNOSTICS_SET_ON, 139

ECHO_OUTPUT, 173

ENTERs, 139

ERRORs, 148

EXITS, 158

FLAG_ERROR_C, 167

FLAG_ERROR_VS, 167

FLAG_WARNING_C, 167

FLAG_WARNING_VS, 168

MAX_OUTPUT_LINES, 173

OP_STRING, 173

OUTPUT_SET_OFF, 168

OUTPUT_SET_ON, 168

ROUTINE_STACK, 174

TIMING, 174

TIMING_ALL_SUBROUTINES, 174

TIMING_FILE_OPEN, 174
 TIMING_FROM_SUBROUTINE, 174
 TIMING_ROUTINE_LIST, 175
 TIMING_SET_OFF, 168
 TIMING_SET_ON, 169
 TIMING_SUMMARY, 175
 TIMING_SUMMARY_OUTPUT, 169
 WRITE_STR, 169
 BASE_ROUTINES::DiagnosticTypes, 26
 BASE_ROUTINES::FileUnits, 22
 BASE_ROUTINES::FLAG_ERROR, 687
 FLAG_ERROR_C, 687
 FLAG_ERROR_VS, 687
 BASE_ROUTINES::FLAG_WARNING, 689
 FLAG_WARNING_C, 689
 FLAG_WARNING_VS, 689
 BASE_ROUTINES::interface, 690
 CPUTIMER, 690
 BASE_ROUTINES::OutputType, 19
 BASE_ROUTINES::ROUTINE_LIST_ITEM_TYPE, 691
 NAME, 691
 NEXT_ROUTINE, 691
 NUMBER_OF_INVOCATIONS, 692
 TOTAL_EXCLUSIVE_CPU_TIME, 692
 TOTAL_EXCLUSIVE_SYSTEM_TIME, 692
 TOTAL_INCLUSIVE_CPU_TIME, 692
 TOTAL_INCLUSIVE_SYSTEM_TIME, 692
 BASE_ROUTINES::ROUTINE_LIST_TYPE, 693
 HEAD, 693
 BASE_ROUTINES::ROUTINE_STACK_ITEM_TYPE, 694
 DIAGNOSTICS, 694
 EXCLUSIVE_CPU_TIME, 694
 EXCLUSIVE_SYSTEM_TIME, 695
 INCLUSIVE_CPU_TIME, 695
 INCLUSIVE_SYSTEM_TIME, 695
 NAME, 695
 PREVIOUS_ROUTINE, 695
 ROUTINE_LIST_ITEM, 695
 TIMING, 695
 BASE_ROUTINES::ROUTINE_STACK_TYPE, 697
 STACK_POINTER, 697
 BASE_ROUTINES::TimingTypes, 28
 BASE_ROUTINES_DiagnosticTypes
 ALL_DIAG_TYPE, 26
 FROM_DIAG_TYPE, 26
 IN_DIAG_TYPE, 26
 BASE_ROUTINES_FileUnits
 DIAGNOSTICS_FILE_UNIT, 23
 ECHO_FILE_UNIT, 23
 IO1_FILE_UNIT, 23
 IO2_FILE_UNIT, 23
 IO3_FILE_UNIT, 23
 IO4_FILE_UNIT, 24
 IO5_FILE_UNIT, 24
 LEARN_FILE_UNIT, 24
 OPEN_COMFILE_UNIT, 24
 START_READ_COMFILE_UNIT, 24
 STOP_READ_COMFILE_UNIT, 25
 TEMPORARY_FILE_UNIT, 25
 TIMING_FILE_UNIT, 25
 BASE_ROUTINES_FINALISE
 BASE_ROUTINES, 138
 BASE_ROUTINES_INITIALISE
 BASE_ROUTINES, 138
 BASE_ROUTINES_OutputType
 DIAGNOSTIC_OUTPUT_TYPE, 19
 ERROR_OUTPUT_TYPE, 20
 GENERAL_OUTPUT_TYPE, 20
 HELP_OUTPUT_TYPE, 20
 TIMING_OUTPUT_TYPE, 21
 BASE_ROUTINES_TimingTypes
 ALL_TIMING_TYPE, 28
 FROM_TIMING_TYPE, 28
 IN_TIMING_TYPE, 28
 BASE_TAG_NUMBER
 TYPES::DISTRIBUTED_MATRIX_CMISS_TYPE, 974
 TYPES::DISTRIBUTED_VECTOR_CMISS_TYPE, 983
 BASES
 TYPES::BASIS_FUNCTIONS_TYPE, 946
 TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE, 1093
 BASIS
 TYPES::DOMAIN_ELEMENT_TYPE, 997
 TYPES::DOMAIN_FACE_TYPE, 1002
 TYPES::DOMAIN_LINE_TYPE, 1008
 TYPES::GENERATED_MESH_REGULAR_TYPE, 1120
 TYPES::MESH_ELEMENT_TYPE, 1135
 TYPES::QUADRATURE_TYPE, 1167
 BASIS_FINISHED
 TYPES::BASIS_TYPE, 951
 BINARY_FILE, 176
 BINARY_FILE_READABLE, 188
 BINARY_FILE_USED, 188
 BINARY_FILE_WRITABLE, 188
 CLOSE_BINARY_FILE, 177
 CLOSE_CMISS_BINARY_FILE, 177
 CMISS_BINARY_FILE_HEADER, 188
 CMISS_BINARY_HISTORY_FILE, 188
 CMISS_BINARY_IDENTITY, 188
 CMISS_BINARY_IDENTITY_HEADER, 188

CMISS_BINARY_MACHINE_HEADER, 188
CMISS_BINARY_MATRIX_FILE, 188
CMISS_BINARY_SIGNAL_FILE, 189
FILE_BEGINNING, 189
FILE_CHANGE_ENDIAN, 189
FILE_CURRENT, 189
FILE_END, 189
FILE_SAME_ENDIAN, 189
INQUIRE_EOF_BINARY_FILE, 178
INQUIRE_OPEN_BINARY_FILE, 178
MAX_NUM_BINARY_FILES, 189
OPEN_BINARY_FILE, 178
OPEN_CMISS_BINARY_FILE, 178
READ_BINARY_FILE_CHARACTER, 179
READ_BINARY_FILE_DP, 179
READ_BINARY_FILE_DP1, 179
READ_BINARY_FILE_DPC, 179
READ_BINARY_FILE_DPC1, 180
READ_BINARY_FILE_INTG, 180
READ_BINARY_FILE_INTG1, 180
READ_BINARY_FILE_LINTG, 180
READ_BINARY_FILE_LINTG1, 180
READ_BINARY_FILE_LOGICAL, 181
READ_BINARY_FILE_LOGICAL1, 181
READ_BINARY_FILE_SINTG, 181
READ_BINARY_FILE_SINTG1, 181
READ_BINARY_FILE_SP, 182
READ_BINARY_FILE_SP1, 182
READ_BINARY_FILE_SPC, 182
READ_BINARY_FILE_SPC1, 182
READ_BINARY_TAG_HEADER, 182
RESET_BINARY_NUMBER_TAGS, 183
SET_BINARY_FILE, 183
SKIP_BINARY_FILE, 183
SKIP_BINARY_TAGS, 183
SKIP_CM_BINARY_HEADER, 183
WRITE_BINARY_FILE_CHARACTER, 184
WRITE_BINARY_FILE_DP, 184
WRITE_BINARY_FILE_DP1, 184
WRITE_BINARY_FILE_DPC, 184
WRITE_BINARY_FILE_DPC1, 185
WRITE_BINARY_FILE_INTG, 185
WRITE_BINARY_FILE_INTG1, 185
WRITE_BINARY_FILE_LINTG, 185
WRITE_BINARY_FILE_LINTG1, 185
WRITE_BINARY_FILE_LOGICAL, 186
WRITE_BINARY_FILE_LOGICAL1, 186
WRITE_BINARY_FILE_SINTG, 186
WRITE_BINARY_FILE_SINTG1, 186
WRITE_BINARY_FILE_SP, 187
WRITE_BINARY_FILE_SP1, 187
WRITE_BINARY_FILE_SPC, 187
WRITE_BINARY_FILE_SPC1, 187
WRITE_BINARY_TAG_HEADER, 187
BINARY_FILE::BINARY_FILE_INFO_TYPE, 698
ACCESS_TYPE, 698
BINARY_FILE_REVISION, 698
CHAR_FORMAT, 698
CHARACTER_SIZE, 698
DP_FORMAT, 698
DP_REAL_SIZE, 699
DPC_REAL_SIZE, 699
ENDIAN_TYPE, 699
FILE_NAME, 699
FILE_NUMBER, 699
INT_FORMAT, 699
INTEGER_SIZE, 699
LINTEGER_SIZE, 699
LOGICAL_SIZE, 699
MACHINE_TYPE, 699
OS_TYPE, 699
SINTEGER_SIZE, 700
SP_FORMAT, 700
SP_REAL_SIZE, 700
SPC_REAL_SIZE, 700
BINARY_FILE::BINARY_FILE_TYPE, 701
FILE_INFORMATION, 701
BINARY_FILE::BINARY_TAG_TYPE, 702
HEADER, 702
INDEX, 702
NUM_BYTES, 702
NUM_HEADER_BYTES, 702
NUM_SUBTAGS, 702
BINARY_FILE::interface, 703
BINARYCLOSEFILE, 703
BINARYOPENFILE, 703
BINARYSETFILE, 703
BINARYSKIPFILE, 703
ISBINARFILEOPEN, 703
ISENDBINARFILE, 704
BINARY_FILE::READ_BINARY_FILE, 705
READ_BINARY_FILE_CHARACTER, 705
READ_BINARY_FILE_DP, 705
READ_BINARY_FILE_DP1, 705
READ_BINARY_FILE_DPC, 705
READ_BINARY_FILE_DPC1, 706
READ_BINARY_FILE_INTG, 706
READ_BINARY_FILE_INTG1, 706
READ_BINARY_FILE_LINTG, 706
READ_BINARY_FILE_LINTG1, 706
READ_BINARY_FILE_LOGICAL, 706
READ_BINARY_FILE_LOGICAL1, 706
READ_BINARY_FILE_SINTG, 707
READ_BINARY_FILE_SINTG1, 707
READ_BINARY_FILE_SP, 707
READ_BINARY_FILE_SP1, 707
READ_BINARY_FILE_SPC, 707
READ_BINARY_FILE_SPC1, 707

READ_BINARY_FILE_SPC, 707
 READ_BINARY_FILE_SPC1, 707
BINARY_FILE::WRITE_BINARY_FILE, 709
 WRITE_BINARY_FILE_CHARACTER, 709
 WRITE_BINARY_FILE_DP, 709
 WRITE_BINARY_FILE_DPI, 709
 WRITE_BINARY_FILE_DPC, 709
 WRITE_BINARY_FILE_DPC1, 710
 WRITE_BINARY_FILE_INTG, 710
 WRITE_BINARY_FILE_INTG1, 710
 WRITE_BINARY_FILE_LINTG, 710
 WRITE_BINARY_FILE_LINTG1, 710
 WRITE_BINARY_FILE_LOGICAL, 710
 WRITE_BINARY_FILE_LOGICAL1, 710
 WRITE_BINARY_FILE_SINTG, 711
 WRITE_BINARY_FILE_SINTG1, 711
 WRITE_BINARY_FILE_SP, 711
 WRITE_BINARY_FILE_SP1, 711
 WRITE_BINARY_FILE_SPC, 711
 WRITE_BINARY_FILE_SPC1, 711
binary_file_c.c
 BinaryCloseFile, 1220
 binaryfiles, 1221
 BinaryOpenFile, 1220
 BinaryReadFile, 1220
 BinarySetFile, 1220
 BinarySkipFile, 1220
 BinaryWriteFile, 1221
 CHARTYPE, 1218
 COMPLEXTYPE, 1218
 DOUBLECOMPLEXTYPE, 1219
 DOUBLETYPE, 1219
 FLIPENDIAN, 1219
 FLOATTYPE, 1219
 INTEGERTYPE, 1219
 IsBinaryFileOpen, 1221
 IsEndBinaryFile, 1221
 logical, 1220
 LOGICALTYPE, 1219
 LONGINTTYPE, 1219
 MAXBINFILES, 1219
 QUADRUPLECOMPLEXTYPE, 1219
 QUADRUPLETYP, 1219
 SAMEENDIAN, 1220
 SHORTINTTYPE, 1220
BINARY_FILE_READABLE
 BINARY_FILE, 188
BINARY_FILE_REVISION
 BINARY_FILE::BINARY_FILE_INFO_TYPE, 698
BINARY_FILE_USED
 BINARY_FILE, 188
BINARY_FILE_WRITABLE
 BINARY_FILE, 188
BINARYCLOSEFILE
 BINARY_FILE::interface, 703
BinaryCloseFile
 binary_file_c.c, 1220
binaryfiles
 binary_file_c.c, 1221
BINARYOPENFILE
 BINARY_FILE::interface, 703
BinaryOpenFile
 binary_file_c.c, 1220
BinaryReadFile
 binary_file_c.c, 1220
BINARYSETFILE
 BINARY_FILE::interface, 703
BinarySetFile
 binary_file_c.c, 1220
BINARYSKIPFILE
 BINARY_FILE::interface, 703
BinarySkipFile
 binary_file_c.c, 1220
BinaryWriteFile
 binary_file_c.c, 1221
BLAS, 190
BLAS::interface, 713
 DASUM, 713
 DAXPY, 713
 DCOPY, 713
 DDOT, 713
 DNRM2, 714
 DROT, 714
 DROTG, 714
 DSCAL, 714
 DTRSV, 714
 SASUM, 714
 SAXPY, 714
 SCOPY, 714
 SDOT, 715
 SNRM2, 715
 SROT, 715
 SROTG, 715
 SSCAL, 715
 STRSV, 715
BLOCK_LENGTHS
 COMP_ENVIRONMENT::MPI_COMPUTATIONAL_NODE_TYPE, 739
BOUNDARY_CONDITIONS
 TYPES::EQUATIONS_SET_FIXED_CONDITIONS_TYPE, 1056
BOUNDARY_LIST
 TYPES::DOMAIN_MAPPING_TYPE, 1013
BUBBLE_SORT_DP
 SORTING, 642
 SORTING::BUBBLE_SORT, 920

BUBBLE_SORT_INTG
 SORTING, 642
 SORTING::BUBBLE_SORT, 920

BUBBLE_SORT_SP
 SORTING, 642
 SORTING::BUBBLE_SORT, 920

C2FSTRING
 F90C, 309

char_auto
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::char, 821

char_fixed
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::char, 821

CHAR_FORMAT
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 698

CHARACTER_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 698
 MACHINE_CONSTANTS, 480

CHARACTER_TO_LOWERCASE_C
 STRINGS, 649
 STRINGS::CHARACTER_TO_-
 LOWERCASE, 923

CHARACTER_TO_LOWERCASE_VS
 STRINGS, 649
 STRINGS::CHARACTER_TO_-
 LOWERCASE, 923

CHARACTER_TO_UPPERCASE_C
 STRINGS, 649
 STRINGS::CHARACTER_TO_-
 UPPERCASE, 924

CHARACTER_TO_UPPERCASE_VS
 STRINGS, 650
 STRINGS::CHARACTER_TO_-
 UPPERCASE, 924

chars
 ISO_VARYING_STRING::VARYING_-
 STRING, 846

CHARTYPE
 binary_file_c.c, 1218

CL
 CLASSICAL_FIELD_ROUTINES, 191

CLASS
 TYPES::EQUATIONS_SET_TYPE, 1065
 TYPES::PROBLEM_TYPE, 1160

CLASSICAL_FIELD_EQUATIONS_SET_-
 FINITE_ELEMENT_CALCULATE
 CLASSICAL_FIELD_ROUTINES, 192

CLASSICAL_FIELD_EQUATIONS_SET_SETUP
 CLASSICAL_FIELD_ROUTINES, 192

CLASSICAL_FIELD_PROBLEM_CLASS_-
 TYPE_SET
 CLASSICAL_FIELD_ROUTINES, 193

CLASSICAL_FIELD_SETUP
 CLASSICAL_FIELD_ROUTINES, 194

CLASSICAL_FIELD_ROUTINES, 191
 CL, 191

CLASSICAL_FIELD_EQUATIONS_SET_-
 FINITE_ELEMENT_CALCULATE,
 192

CLASSICAL_FIELD_EQUATIONS_SET_-
 SETUP, 192

CLASSICAL_FIELD_PROBLEM_CLASS_-
 TYPE_SET, 193

CLASSICAL_FIELD_PROBLEM_SETUP,
 194

CLOSE_BINARY_FILE
 BINARY_FILE, 177

CLOSE_CMISS_BINARY_FILE
 BINARY_FILE, 177

CMISS, 195
 CMISS_BUILD_VERSION, 196
 CMISS_FINALISE, 195
 CMISS_INITIALISE, 195
 CMISS_MAJOR_VERSION, 196
 CMISS_MINOR_VERSION, 196
 CMISS_WRITE_ERROR, 196
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 979
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 991

CMISS_BINARY_FILE_HEADER
 BINARY_FILE, 188

CMISS_BINARY_HISTORY_FILE
 BINARY_FILE, 188

CMISS_BINARY_IDENTITY
 BINARY_FILE, 188

CMISS_BINARY_IDENTITY_HEADER
 BINARY_FILE, 188

CMISS_BINARY_MACHINE_HEADER
 BINARY_FILE, 188

CMISS_BINARY_MATRIX_FILE
 BINARY_FILE, 188

CMISS_BINARY_SIGNAL_FILE
 BINARY_FILE, 189

CMISS_BUILD_VERSION
 CMISS, 196

CMISS_FINALISE
 CMISS, 195

CMISS_INITIALISE
 CMISS, 195

CMISS_MAJOR_VERSION
 CMISS, 196

CMISS_MINOR_VERSION

CMISS, 196
 CMISS_MPI, 197
 MPI_ERROR_CHECK, 197
 CMISS_PARMETIS, 198
 PARMETIS_PARTKWAY, 198
 PARMETIS_PARTMESHKWAY, 198
 CMISS_PARMETIS::interface, 716
 ParMETIS_V3_PartK, 716
 ParMETIS_V3_PartMeshKway, 716
 CMISS_PETSC, 200
 PETSC_ERRORHANDLING_SET_OFF, 205
 PETSC_ERRORHANDLING_SET_ON, 205
 PETSC_FINALIZE, 206
 PETSC_HANDLE_ERROR, 239
 PETSC_INITIALIZE, 206
 PETSC_ISDESTROY, 206
 PETSC_ISFINALISE, 207
 PETSC_ISINITIALISE, 207
 PETSC_ISLOCALTOGLOBALMAPPINGAPPLY,
 207
 PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS,
 208
 PETSC_ISLOCALTOGLOBALMAPPINGCREATE,
 208
 PETSC_ISLOCALTOGLOBALMAPPINGDESTROY,
 209
 PETSC_ISLOCALTOGLOBALMAPPINGFINALISE,
 209
 PETSC_ISLOCALTOGLOBALMAPPINGINITIALISE,
 209
 PETSC_KSPCREATE, 210
 PETSC_KSPDESTROY, 210
 PETSC_KSPFINALISE, 211
 PETSC_KSPGETCONVERGEDREASON,
 211
 PETSC_KSPGETITERATIONNUMBER, 211
 PETSC_KSPGETPC, 212
 PETSC_KSPGETRESIDUALNORM, 212
 PETSC_KSPINITIALISE, 213
 PETSC_KSPSETFROMOPTIONS, 213
 PETSC_KSPSETOPERATORS, 213
 PETSC_KSPSETTOLERANCES, 214
 PETSC_KSPSETTYPE, 214
 PETSC_KSPSETUP, 215
 PETSC_KSPSOLVE, 215
 PETSC_LOGPRINTSUMMARY, 216
 PETSC_MATASSEMBLYBEGIN, 216
 PETSC_MATASSEMBLYEND, 216
 PETSC_MATCREATE, 217
 PETSC_MATCREATEMPIAIJ, 217
 PETSC_MATCREATEMPIDENSE, 218
 PETSC_MATCREATESEQQAIJ, 218
 PETSC_MATCREATESEQDENSE, 219
 PETSC_MATDESTROY, 219
 PETSC_MATFINALISE, 220
 PETSC_MATGETARRAY, 220
 PETSC_MATGETOWNERSHIPRANGE, 220
 PETSC_MATGETVALUES, 221
 PETSC_MATINITIALISE, 221
 PETSC_MATRESTOREARRAY, 221
 PETSC_MATSETLOCALTOGLOBALMAPPING,
 222
 PETSC_MATSETOPTION, 222
 PETSC_MATSETSIZES, 223
 PETSC_MATSETVALUE, 223
 PETSC_MATSETVALUELOCAL, 223
 PETSC_MATSETVALUES, 224
 PETSC_MATSETVALUESLOCAL, 224
 PETSC_MATVIEW, 225
 PETSC_MATZEROENTRIES, 225
 PETSC_PCFINALISE, 226
 PETSC_PCINITIALISE, 226
 PETSC_PCSETTYPE, 226
 PETSC_VECASSEMBLYBEGIN, 227
 PETSC_VECASSEMBLYEND, 227
 PETSC_VECCREATE, 228
 PETSC_VECCREATEGHOST, 228
 PETSC_VECCREATEGHOSTWITHARRAY,
 228
 PETSC_VECCREATEMPI, 229
 PETSC_VECCREATEMPIWITHARRAY,
 229
 PETSC_VECCREATESEQ, 230
 PETSC_VECCREATESEQWITHARRAY,
 230
 PETSC_VECDESTROY, 231
 PETSC_VECDUPLICATE, 231
 PETSC_VECFINALISE, 231
 PETSC_VECGETARRAY, 232
 PETSC_VECGETARRAYF90, 232
 PETSC_VECGETLOCALSIZE, 232
 PETSC_VECGETOWNERSHIPRANGE, 233
 PETSC_VECGETSIZE, 233
 PETSC_VECGETVALUES, 234
 PETSC_VECGHOSTGETLOCALFORM,
 234
 PETSC_VECGHOSTRESTORELOCALFORM,
 234
 PETSC_VECGHOSTUPDATEBEGIN, 235
 PETSC_VECGHOSTUPDATEEND, 235
 PETSC_VEGINITIALISE, 236
 PETSC_VCRESTOREARRAY, 236
 PETSC_VCRESTOREARRAYF90, 236
 PETSC_VECSET, 237
 PETSC_VECSETFROMOPTIONS, 237
 PETSC_VECSETLOCALTOGLOBALMAPPING,
 237
 PETSC_VECSETSIZES, 238

PETSC_VECSETVALUES, 238
PETSC_VECSETVALUESLOCAL, 239
PETSC_VECVIEW, 239
CMISS_PETSC::interface, 717
ISDestroy, 718
ISLocalToGlobalMappingApply, 718
ISLocalToGlobalMappingApplyIS, 719
ISLocalToGlobalMappingCreate, 719
ISLocalToGlobalMappingDestroy, 719
KSPCreate, 719
KSPDestroy, 719
KSPGetConvergedReason, 719
KSPGetIterationNumber, 719
KSPGetPC, 719
KSPGetResidualNorm, 719
KSPSetFromOptions, 719
KSPSetOperators, 720
KSPSetTolerances, 720
KSPSetType, 720
KSPSetUp, 720
KSPSolve, 720
MatAssemblyBegin, 720
MatAssemblyEnd, 720
MatCreate, 720
MatCreateMPIAIJ, 720
MatCreateMPIDense, 720
MatCreateSeqAIJ, 721
MatCreateSeqDense, 721
MatDestroy, 721
MatGetArray, 721
MatGetArrayF90, 721
MatGetOwnershipRange, 721
MatGetValues, 721
MatRestoreArray, 721
MatRestoreArrayF90, 721
MatSetLocalToGlobalMapping, 721
MatSetOption, 722
MatSetSizes, 722
MatSetValue, 722
MatSetValueLocal, 722
MatSetValues, 722
MatSetValuesLocal, 722
MatView, 722
MatZeroEntries, 722
PCsetType, 722
PetscFinalize, 722
PetscInitialize, 723
PetscLogPrintSummary, 723
SNESCreate, 723
SNESDestroy, 723
SNESSetFromOptions, 723
SNESSetFunction, 723
SNESSetType, 723
SNESSolve, 723
VecAssemblyBegin, 723
VecAssemblyEnd, 723
VecCreate, 723
VecCreateGhost, 724
VecCreateGhostWithArray, 724
VecCreateMPI, 724
VecCreateMPIWithArray, 724
VecCreateSeq, 724
VecCreateSeqWithArray, 724
VecDestroy, 724
VecDuplicate, 724
VecGetArray, 724
VecGetArrayF90, 724
VecGetLocalSize, 725
VecGetOwnershipRange, 725
VecGetSize, 725
VecGetValues, 725
VecGhostGetLocalForm, 725
VecGhostRestoreLocalForm, 725
VecGhostUpdateBegin, 725
VecGhostUpdateEnd, 725
VecRestoreArray, 725
VecRestoreArrayF90, 725
VecSet, 726
VecSetFromOptions, 726
VecSetLocalToGlobalMapping, 726
VecSetSizes, 726
VecSetValue, 726
VecSetValueLocal, 726
VecView, 726
CMISS_PETSC_TYPES, 240
CMISS_PETSC_TYPES::PETSC_IS_TYPE, 727
CMISS_PETSC_TYPES::PETSC_-
ISLOCALTOGLOBALMAPPING_-
TYPE, 728
CMISS_PETSC_TYPES::PETSC_KSP_TYPE,
729
CMISS_PETSC_TYPES::PETSC_MAT_TYPE,
730
CMISS_PETSC_TYPES::PETSC_PC_TYPE, 731
CMISS_PETSC_TYPES::PETSC_SNES_TYPE,
732
CMISS_PETSC_TYPES::PETSC_VEC_TYPE,
733
CMISS_VECTOR
TYPES::DISTRIBUTED_VECTOR_-
TRANSFER_TYPE, 988
CMISS_WRITE_ERROR
CMISS, 196
CO
COORDINATE_ROUTINES, 249, 250
COLLAPSED_XI
TYPES::BASIS_TYPE, 951
COLOUR

TREES::TREE_NODE_TYPE, 942
 COLUMN_DOF
 TYPES::VARIABLE_TO_EQUATIONS_-
 COLUMN_MAP_TYPE, 1201
 COLUMN_DOFS
 TYPES::ELEMENT_MATRIX_TYPE, 1028
 COLUMN_DOFS_MAPPING
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1047
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1184
 COLUMN_DOMAIN_MAPPING
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 979
 COLUMN_INDICES
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 976
 TYPES::MATRIX_TYPE, 1130
 COLUMN_NUMBERS
 TYPES::VARIABLE_TO_SOLVER_COL_-
 MAP_TYPE, 1204
 COLUMN_TO_DOF_MAP
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1047
 COMP_ENVIRONMENT, 241
 COMPUTATIONAL_ENVIRONMENT, 246
 COMPUTATIONAL_ENVIRONMENT_-
 FINALISE, 242
 COMPUTATIONAL_ENVIRONMENT_-
 INITIALISE, 242
 COMPUTATIONAL_NODE_FINALISE, 243
 COMPUTATIONAL_NODE_INITIALISE,
 243
 COMPUTATIONAL_NODE_MPI_TYPE_-
 FINALISE, 244
 COMPUTATIONAL_NODE_MPI_TYPE_-
 INITIALISE, 244
 COMPUTATIONAL_NODE_NUMBER_-
 GET, 244
 COMPUTATIONAL_NODES_NUMBER_-
 GET, 245
 MPI_COMPUTATIONAL_NODE_TYPE_-
 DATA, 246
 COMP_ENVIRONMENT::CACHE_TYPE, 734
 NUMBER_LEVELS, 734
 SIZE, 734
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 ENVIRONMENT_TYPE, 735
 COMPUTATIONAL_NODES, 735
 MPI_COMM, 735
 MY_COMPUTATIONAL_NODE_-
 NUMBER, 735
 NUMBER_COMPUTATIONAL_NODES,
 736
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 NODE_TYPE, 737
 NODE_NAME, 737
 NODE_NAME_LENGTH, 737
 NUMBER_PROCESSORS, 737
 RANK, 737
 COMP_ENVIRONMENT::MPI_-
 COMPUTATIONAL_NODE_TYPE,
 739
 BLOCK_LENGTHS, 739
 DISPLACEMENTS, 739
 MPI_TYPE, 739
 NUM_BLOCKS, 739
 TYPES, 740
 COMPLEXTYPE
 binary_file_c.c, 1218
 COMPONENT_NUMBER
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113
 COMPONENTS
 FIELD_IO_ROUTINES::FIELD_IO_INFO_-
 SET, 763
 TYPES::FIELD_VARIABLE_TYPE, 1117
 COMPUTATIONAL_ENVIRONMENT
 COMP_ENVIRONMENT, 246
 COMPUTATIONAL_ENVIRONMENT_-
 FINALISE
 COMP_ENVIRONMENT, 242
 COMPUTATIONAL_ENVIRONMENT_-
 INITIALISE
 COMP_ENVIRONMENT, 242
 COMPUTATIONAL_NODE_FINALISE
 COMP_ENVIRONMENT, 243
 COMPUTATIONAL_NODE_INITIALISE
 COMP_ENVIRONMENT, 243
 COMPUTATIONAL_NODE_MPI_TYPE_-
 FINALISE
 COMP_ENVIRONMENT, 244
 COMPUTATIONAL_NODE_MPI_TYPE_-
 INITIALISE
 COMP_ENVIRONMENT, 244
 COMPUTATIONAL_NODE_NUMBER_GET
 COMP_ENVIRONMENT, 244
 COMPUTATIONAL_NODES
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 ENVIRONMENT_TYPE, 735
 COMPUTATIONAL_NODES_NUMBER_GET
 COMP_ENVIRONMENT, 245
 CONSTANT_DOF2PARAM_MAP
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1083
 CONSTANT_PARAM2DOF_MAP
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097

CONTROL
 TYPES::PROBLEM_TYPE, [1160](#)

CONTROL_FINISHED
 TYPES::PROBLEM_CONTROL_TYPE, [1155](#)

CONTROL_TYPE
 TYPES::PROBLEM_CONTROL_TYPE, [1155](#)

COORDINAT
 COORDINATE_ROUTINES, [260](#)

COORDINATE_CONVERT_FROM_RC_DP
 COORDINATE_ROUTINES, [250](#)

COORDINATE_-
 ROUTINES::COORDINATE_-
 CONVERT_FROM_RC, [741](#)

COORDINATE_CONVERT_FROM_RC_SP
 COORDINATE_ROUTINES, [250](#)

COORDINATE_-
 ROUTINES::COORDINATE_-
 CONVERT_FROM_RC, [741](#)

COORDINATE_CONVERT_TO_RC_DP
 COORDINATE_ROUTINES, [250](#)

COORDINATE_-
 ROUTINES::COORDINATE_-
 CONVERT_TO_RC, [742](#)

COORDINATE_CONVERT_TO_RC_SP
 COORDINATE_ROUTINES, [251](#)

COORDINATE_-
 ROUTINES::COORDINATE_-
 CONVERT_TO_RC, [742](#)

COORDINATE_CYCLINDRICAL_POLAR_-
 TYPE

COORINDATE_ROUTINES_-
 CoordinateSystemTypes, [30](#)

COORDINATE_DELTA_CALCULATE_DP
 COORDINATE_ROUTINES, [251](#)

COORDINATE_-
 ROUTINES::COORDINATE_DELTA_-
 CALCULATE, [743](#)

COORDINATE_DERIVATIVE_CONVERT_TO_-
 RC_DP

COORDINATE_-
 ROUTINES::COORDINATE_-
 DERIVATIVE_CONVERT_TO_RC, [744](#)

COORDINATE_DERIVATIVE_CONVERT_TO_-
 RC_SP

COORDINATE_-
 ROUTINES::COORDINATE_-
 DERIVATIVE_CONVERT_TO_RC, [744](#)

COORDINATE_DERIVATIVE_NORM
 COORDINATE_ROUTINES, [251](#)

COORDINATE_INTERPOLATION_ADJUST
 COORDINATE_ROUTINES, [252](#)

COORDINATE_PARAMETERS_ADJUST
 COORDINATE_ROUTINES, [252](#)

COORDINATE_JACOBIAN_AREA_TYPE
 COORDINATE_ROUTINES_JacobianType, [36](#)

COORDINATE_JACOBIAN_LINE_TYPE
 COORDINATE_ROUTINES_JacobianType, [36](#)

COORDINATE_JACOBIAN_VOLUME_TYPE
 COORDINATE_ROUTINES_JacobianType, [36](#)

COORDINATE_METRICS_CALCULATE
 COORDINATE_ROUTINES, [252](#)

COORDINATE_NO_RADIAL_-
 INTERPOLATION_TYPE

COORDINATE_ROUTINES_-
 RadialInterpolations, [34](#)

COORDINATE_OBLATE_SPHEROIDAL_TYPE
 COORDINATE_ROUTINES_-
 CoordinateSystemTypes, [31](#)

COORDINATE_PROLATE_SPHEROIDAL_-
 TYPE

COORINDATE_ROUTINES_-
 CoordinateSystemTypes, [31](#)

COORDINATE_RADIAL_CUBED_-
 INTERPOLATION_TYPE

COORDINATE_ROUTINES_-
 RadialInterpolations, [34](#)

COORDINATE_RADIAL_INTERPOLATION_-
 TYPE

COORDINATE_ROUTINES_-
 RadialInterpolations, [35](#)

COORDINATE_RADIAL_SQUARED_-
 INTERPOLATION_TYPE

COORDINATE_ROUTINES_-
 RadialInterpolations, [35](#)

COORDINATE_RECTANGULAR_-
 CARTESIAN_TYPE

COORINDATE_ROUTINES_-
 CoordinateSystemTypes, [32](#)

COORDINATE_ROUTINES, [247](#)
 _SYSTEM_TYPE_STRING, [260](#)

CO, [249](#), [250](#)

COORDINAT, [260](#)

COORDINATE_CONVERT_FROM_RC_DP, [250](#)

COORDINATE_CONVERT_FROM_RC_SP, [250](#)

COORDINATE_CONVERT_TO_RC_DP, [250](#)

COORDINATE_CONVERT_TO_RC_SP, [251](#)

COORDINATE_DELTA_CALCULATE_DP,
 251
 COORDINATE_DERIVATIVE_NORM, 251
 COORDINATE_INTERPOLATION_-
 ADJUST, 252
 COORDINATE_INTERPOLATION_-
 PARAMETERS_ADJUST, 252
 COORDINATE_METRICS_CALCULATE,
 252
 COORDINATE_SYSTEM_CREATE_-
 FINISH, 253
 COORDINATE_SYSTEM_CREATE_-
 START, 253
 COORDINATE_SYSTEM_DESTROY_-
 NUMBER, 253
 COORDINATE_SYSTEM_DESTROY_PTR,
 254
 COORDINATE_SYSTEM_DIMENSION_-
 GET, 254
 COORDINATE_SYSTEM_DIMENSION_-
 SET_NUMBER, 254
 COORDINATE_SYSTEM_DIMENSION_-
 SET_PTR, 254
 COORDINATE_SYSTEM_FOCUS_GET,
 254
 COORDINATE_SYSTEM_FOCUS_SET_-
 NUMBER, 255
 COORDINATE_SYSTEM_FOCUS_SET_-
 PTR, 255
 COORDINATE_SYSTEM_NORMAL_-
 CALCULATE, 255
 COORDINATE_SYSTEM_ORIENTATION_-
 SET_NUMBER, 256
 COORDINATE_SYSTEM_ORIENTATION_-
 SET_PTR, 256
 COORDINATE_SYSTEM_ORIGIN_SET_-
 NUMBER, 256
 COORDINATE_SYSTEM_ORIGIN_SET_-
 PTR, 256
 COORDINATE_SYSTEM_RADIAL_-
 INTERPOLATION_TYPE_GET, 257
 COORDINATE_SYSTEM_RADIAL_-
 INTERPOLATION_TYPE_SET_-
 NUMBER, 257
 COORDINATE_SYSTEM_RADIAL_-
 INTERPOLATION_TYPE_SET_PTR,
 257
 COORDINATE_SYSTEM_TYPE_GET, 257
 COORDINATE_SYSTEM_TYPE_SET_-
 NUMBER, 257
 COORDINATE_SYSTEM_TYPE_SET_PTR,
 258
 COORDINATE_SYSTEM_USER_-
 NUMBER_FIND, 258

COORDINATE_SYSTEMS, 260
 COORDINATE_SYSTEMS_FINALISE, 258
 COORDINATE_SYSTEMS_INITIALISE,
 259
 D2ZX_DP, 259
 DXZ_DP, 259
 DZX_DP, 260
 GLOBAL_COORDINATE_SYSTEM, 260
 COORDINATE_ROUTINES::COORDINATE_-
 CONVERT_FROM_RC, 741
 COORDINATE_CONVERT_FROM_RC_DP,
 741
 COORDINATE_CONVERT_FROM_RC_SP,
 741
 COORDINATE_ROUTINES::COORDINATE_-
 CONVERT_TO_RC, 742
 COORDINATE_CONVERT_TO_RC_DP,
 742
 COORDINATE_CONVERT_TO_RC_SP, 742
 COORDINATE_ROUTINES::COORDINATE_-
 DELTA_CALCULATE, 743
 COORDINATE_DELTA_CALCULATE_DP,
 743
 COORDINATE_ROUTINES::COORDINATE_-
 DERIVATIVE_CONVERT_TO_RC,
 744
 COORDINATE_DERIVATIVE_CONVERT_-
 TO_RC_DP, 744
 COORDINATE_DERIVATIVE_CONVERT_-
 TO_RC_SP, 744
 COORDINATE_ROUTINES::COORDINATE_-
 SYSTEM_DESTROY, 745
 COORDINATE_SYSTEM_DESTROY_-
 NUMBER, 745
 COORDINATE_SYSTEM_DESTROY_PTR,
 745
 COORDINATE_ROUTINES::COORDINATE_-
 SYSTEM_DIMENSION_SET, 746
 COORDINATE_SYSTEM_DIMENSION_-
 SET_NUMBER, 746
 COORDINATE_SYSTEM_DIMENSION_-
 SET_PTR, 746
 COORDINATE_ROUTINES::COORDINATE_-
 SYSTEM_FOCUS_SET, 747
 COORDINATE_SYSTEM_FOCUS_SET_-
 NUMBER, 747
 COORDINATE_SYSTEM_FOCUS_SET_-
 PTR, 747
 COORDINATE_ROUTINES::COORDINATE_-
 SYSTEM_ORIENTATION_SET, 748
 COORDINATE_SYSTEM_ORIENTATION_-
 SET_NUMBER, 748
 COORDINATE_SYSTEM_ORIENTATION_-
 SET_PTR, 748

COORDINATE_ROUTINES::COORDINATE_SYSTEM_ORIGIN_SET, 749
COORDINATE_SYSTEM_ORIGIN_SET_NUMBER, 749
COORDINATE_SYSTEM_ORIGIN_SET_PTR, 749
COORDINATE_ROUTINES::COORDINATE_SYSTEM_PTR_TYPE, 750
PTR, 750
COORDINATE_ROUTINES::COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET, 751
COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_NUMBER, 751
COORDINATE_SYSTEM_RADIAL_INTERPOLATION_TYPE_SET_PTR, 751
COORDINATE_ROUTINES::COORDINATE_SYSTEM_TYPE_SET, 752
COORDINATE_SYSTEM_TYPE_SET_NUMBER, 752
COORDINATE_SYSTEM_TYPE_SET_PTR, 752
COORDINATE_ROUTINES::COORDINATE_SYSTEMS_TYPE, 753
COORDINATE_SYSTEMS, 753
NUMBER_OF_COORDINATE_SYSTEMS, 753
COORDINATE_ROUTINES::CoordinateSystemType, 30
COORDINATE_ROUTINES::D2ZX, 754
D2ZX_DP, 754
COORDINATE_ROUTINES::DXZ, 755
DXZ_DP, 755
COORDINATE_ROUTINES::DZX, 756
DZX_DP, 756
COORDINATE_ROUTINES::JacobianType, 36
COORDINATE_ROUTINES::RadialInterpolations, 34
COORDINATE_ROUTINES_JacobianType
COORDINATE_JACOBIAN_AREA_TYPE, 36
COORDINATE_JACOBIAN_LINE_TYPE, 36
COORDINATE_JACOBIAN_VOLUME_TYPE, 36
COORDINATE_ROUTINES_RadialInterpolations
COORDINATE_NO_RADIAL_INTERPOLATION_TYPE, 34
COORDINATE_RADIAL_CUBED_INTERPOLATION_TYPE, 34
COORDINATE_RADIAL_INTERPOLATION_TYPE, 35
COORDINATE_RADIAL_SQUARED_INTERPOLATION_TYPE, 35
COORDINATE_SPHERICAL_POLAR_TYPE
COORINDATE_ROUTINES_CoordinateSystemTypes, 32
COORDINATE_SYSTEM
TYPES::REGION_TYPE, 1171
COORDINATE_SYSTEM_CREATE_FINISH
COORDINATE_ROUTINES, 253
COORDINATE_SYSTEM_CREATE_START
COORDINATE_ROUTINES, 253
COORDINATE_SYSTEM_DESTROY_NUMBER
COORDINATE_ROUTINES, 253
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY, 745
COORDINATE_SYSTEM_DESTROY_PTR
COORDINATE_ROUTINES, 254
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DESTROY, 745
COORDINATE_SYSTEM_DIMENSION_GET
COORDINATE_ROUTINES, 254
COORDINATE_SYSTEM_DIMENSION_SET_NUMBER
COORDINATE_ROUTINES, 254
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET, 746
COORDINATE_SYSTEM_DIMENSION_SET_PTR
COORDINATE_ROUTINES, 254
COORDINATE_ROUTINES::COORDINATE_SYSTEM_DIMENSION_SET, 746
COORDINATE_SYSTEM_FINISHED
TYPES::COORDINATE_SYSTEM_TYPE, 959
COORDINATE_SYSTEM_FOCUS_GET
COORDINATE_ROUTINES, 254
COORDINATE_SYSTEM_FOCUS_SET_NUMBER
COORDINATE_ROUTINES, 255
COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET, 747
COORDINATE_SYSTEM_FOCUS_SET_PTR
COORDINATE_ROUTINES, 255
COORDINATE_ROUTINES::COORDINATE_SYSTEM_FOCUS_SET, 747
COORDINATE_SYSTEM_NORMAL_CALCULATE
COORDINATE_ROUTINES, 255

COORDINATE_SYSTEM_ORIENTATION_-
SET_NUMBER
COORDINATE_ROUTINES, 256
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_ORIENTATION_SET, 748
COORDINATE_SYSTEM_ORIENTATION_-
SET_PTR
COORDINATE_ROUTINES, 256
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_ORIENTATION_SET, 748
COORDINATE_SYSTEM_ORIGIN_SET_-
NUMBER
COORDINATE_ROUTINES, 256
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_ORIGIN_SET, 749
COORDINATE_SYSTEM_ORIGIN_SET_PTR
COORDINATE_ROUTINES, 256
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_ORIGIN_SET, 749
COORDINATE_SYSTEM_RADIAL_-
INTERPOLATION_TYPE_GET
COORDINATE_ROUTINES, 257
COORDINATE_SYSTEM_RADIAL_-
INTERPOLATION_TYPE_SET_-
NUMBER
COORDINATE_ROUTINES, 257
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_RADIAL_-
INTERPOLATION_TYPE_SET, 751
COORDINATE_SYSTEM_RADIAL_-
INTERPOLATION_TYPE_SET_PTR
COORDINATE_ROUTINES, 257
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_RADIAL_-
INTERPOLATION_TYPE_SET, 751
COORDINATE_SYSTEM_TYPE_GET
COORDINATE_ROUTINES, 257
COORDINATE_SYSTEM_TYPE_SET_-
NUMBER
COORDINATE_ROUTINES, 257
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_TYPE_SET, 752
COORDINATE_SYSTEM_TYPE_SET_PTR
COORDINATE_ROUTINES, 258
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEM_TYPE_SET, 752
COORDINATE_SYSTEM_USER_NUMBER_-
FIND
COORDINATE_ROUTINES, 258
COORDINATE_SYSTEMS
COORDINATE_ROUTINES, 260
COORDINATE_-
ROUTINES::COORDINATE_-
SYSTEMS_TYPE, 753
COORDINATE_SYSTEMS_FINALISE
COORDINATE_ROUTINES, 258
COORDINATE_SYSTEMS_INITIALISE
COORDINATE_ROUTINES, 259
COORINDATE_ROUTINES_-
CoordinateSystemTypes
COORDINATE_CYCLINDRICAL_-
POLAR_TYPE, 30
COORDINATE_OBLATE_SPHEROIDAL_-
TYPE, 31
COORDINATE_PROLATE_SPHEROIDAL_-
TYPE, 31
COORDINATE_RECTANGULAR_-
CARTESIAN_TYPE, 32
COORDINATE_SPHERICAL_POLAR_-
TYPE, 32
COUPLING_COEFFICIENTS
TYPES::EQUATIONS_COL_TO_SOLVER_-
COLS_MAP_TYPE, 1031
TYPES::EQUATIONS_ROW_TO_-
SOLVER_ROWS_MAP_TYPE, 1052
TYPES::SOLVER_COL_TO_EQUATIONS_-
MAP_TYPE, 1180
TYPES::SOLVER_ROW_TO_-
EQUATIONS_SET_MAP_TYPE,
1194
TYPES::VARIABLE_TO_SOLVER_COL_-
MAP_TYPE, 1204
CPU_TIMER
TIMER, 665
CPUTIMER
BASE_ROUTINES::interface, 690
TIMER::interface, 941
CPUTimer
timer_c.c, 1380
CREATE_VALUES_CACHE
TYPES::EQUATIONS_MAPPING_TYPE,
1040
TYPES::SOLUTION_MAPPING_TYPE,
1175
CROSS_PRODUCT_DP
MATHS, 485
MATHS::CROSS_PRODUCT, 870
CROSS_PRODUCT_INTG
MATHS, 485
MATHS::CROSS_PRODUCT, 870

CROSS_PRODUCT_SP
 MATHS, 485
 MATHS::CROSS_PRODUCT, 870

CSTRINGLEN
 F90C::interface, 760

CStringLen
 f90c_c.c, 1272

CSTRINGLENGTH
 F90C, 310

D2ZX_DP
 COORDINATE_ROUTINES, 259
 COORDINATE_ROUTINES::D2ZX, 754

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/base_routines.f90, 1211

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/basis_routines.f90, 1217

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/binary_file_c.c, 1218

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/binary_file_f.f90, 1222

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/blas.f90, 1226

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/classical_field_routines.f90,
 1228

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/cmiss.f90, 1230

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/cmiss_mpi.f90, 1231

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/cmiss_parmetis.f90, 1233

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/cmiss_petsc.f90, 1235

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/cmiss_petsc_types.f90, 1242

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/computational_-
 environment.f90, 1244

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/constants.f90, 1247

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/coordinate_routines.f90, 1248

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/distributed_matrix_vector.f90,
 1253

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/domain_mappings.f90, 1254

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/elasticity_routines.f90, 1256

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/electromechanics_routines.f90,
 1257

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/equations_mapping_-
 routines.f90, 1258

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/equations_matrices_-
 routines.f90, 1259

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/equations_set_constants.f90,
 1260

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/equations_set_routines.f90,
 1265

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/f90c_c.c, 1272

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/f90c_f.f90, 1273

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/field_IO_routines.f90, 1275

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/field_routines.f90, 1280

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/finite_element_routines.f90,
 1286

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/fluid_mechanics_routines.f90,
 1288

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/generated_mesh_routines.f90,
 1289

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/input_output.f90, 1291

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/iso_varying_string.f90, 1306

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/kinds.f90, 1310

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/lapack.f90, 1312

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/Laplace_equations_-
 routines.f90, 1314

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/lists.f90, 1316

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/machine_constants_aix.f90,
 1321

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/machine_constants_irix.f90,
 1323

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/machine_constants_linux.f90,
 1324

d:/Users/tyu011/workspace/OpenCMISS-
 trunk/src/machine_constants_vms.f90,
 1325

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/machine_constants_win32.f90, [1326](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/math.f90, [1327](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/matrix_vector.f90, [1330](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/mesh_routines.f90, [1342](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/node_routines.f90, [1349](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_constants.f90, [1351](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/problem_routines.f90, [1355](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/region_routines.f90, [1360](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solution_mapping_routines.f90, [1362](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_matrices_routines.f90, [1363](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/solver_routines.f90, [1365](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/sorting.f90, [1372](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/strings.f90, [1374](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_c.c, [1380](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/timer_f.f90, [1381](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/trees.f90, [1383](#)

d:/Users/tyu011/workspace/OpenCMISS-trunk/src/types.f90, [1386](#)

D_CROSS_PRODUCT_DP
 MATHS, [485](#)
 MATHS::D_CROSS_PRODUCT, [871](#)

D_CROSS_PRODUCT_INTG
 MATHS, [486](#)
 MATHS::D_CROSS_PRODUCT, [871](#)

D_CROSS_PRODUCT_SP
 MATHS, [486](#)
 MATHS::D_CROSS_PRODUCT, [871](#)

DASUM
 BLAS::interface, [713](#)

DATA_DP
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, [976](#)
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, [983](#)
 TYPES::MATRIX_TYPE, [1130](#)

DATA_INTG
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, [983](#)
 TYPES::MATRIX_TYPE, [1130](#)
 TYPES::VECTOR_TYPE, [1206](#)

DATA_L
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, [983](#)
 TYPES::MATRIX_TYPE, [1130](#)
 TYPES::VECTOR_TYPE, [1207](#)

DATA_SIZE
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, [976](#)
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, [983](#)
 TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, [985](#)

DATA_SP
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, [983](#)
 TYPES::MATRIX_TYPE, [1130](#)
 TYPES::VECTOR_TYPE, [1207](#)

DATA_TYPE
 LISTS::LIST_TYPE, [867](#)
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 [980](#)
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, [988](#)
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 [991](#)
 TYPES::MATRIX_TYPE, [1131](#)
 TYPES::VECTOR_TYPE, [1207](#)

DAXPY
 BLAS::interface, [713](#)

DCOPY
 BLAS::interface, [713](#)

DDOT
 BLAS::interface, [713](#)

DECOMPOSITION
 TYPES::DECOMPOSITION_ELEMENTS_-
 TYPE, [963](#)
 TYPES::DECOMPOSITION_LINES_TYPE,
 [967](#)
 TYPES::DECOMPOSITION_TOPOLOGY_-
 TYPE, [969](#)
 TYPES::DOMAIN_TYPE, [1025](#)
 TYPES::FIELD_TYPE, [1109](#)

DECOMPOSITION_ALL_TYPE
 MESH_ROUTINES_DecompositionTypes, [97](#)

DECOMPOSITION_CALCULATED_TYPE
 MESH_ROUTINES_DecompositionTypes, [97](#)

DECOMPOSITION_CREATE_FINISH
 MESH_ROUTINES, [547](#)

DECOMPOSITION_CREATE_START
 MESH_ROUTINES, 547

DECOMPOSITION_DESTROY
 MESH_ROUTINES, 548

DECOMPOSITION_ELEMENT_DOMAIN_-
 CALCULATE
 MESH_ROUTINES, 548

DECOMPOSITION_ELEMENT_DOMAIN_SET
 MESH_ROUTINES, 549

DECOMPOSITION_FINISHED
 TYPES::DECOMPOSITION_TYPE, 971

DECOMPOSITION_MESH_COMPONENT_-
 NUMBER_SET
 MESH_ROUTINES, 549

DECOMPOSITION_NUMBER_OF_DOMAINS_-
 SET
 MESH_ROUTINES, 550

DECOMPOSITION_TYPE
 TYPES::DECOMPOSITION_TYPE, 971

DECOMPOSITION_USER_DEFINED_TYPE
 MESH_ROUTINES_DecompositionTypes, 98

DECOMPOSITIONS
 TYPES::DECOMPOSITION_TYPE, 971
 TYPES::DECOMPOSITIONS_TYPE, 973
 TYPES::MESH_TYPE, 1146

DEGENERATE
 TYPES::BASIS_TYPE, 951

DEPENDENT
 TYPES::EQUATIONS_SET_TYPE, 1065

DEPENDENT_FIELD
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1033

 TYPES::EQUATIONS_SET_DEPENDENT_-
 TYPE, 1055

DEPENDENT_FINISHED
 TYPES::EQUATIONS_SET_DEPENDENT_-
 TYPE, 1055

DEPENDENT_INTERP_PARAMETERS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1033

DEPENDENT_INTERP_POINT
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1033

DEPENDENT_TYPE
 TYPES::FIELD_TYPE, 1109

DERIVATIVE_NUMBERS_IN_LOCAL_LINE
 TYPES::BASIS_TYPE, 951

DERIVATIVE_ORDER_INDEX
 TYPES::BASIS_TYPE, 952

DERIVATIVE_ORDER_INDEX_INV
 TYPES::BASIS_TYPE, 952

DERIVATIVES_IN_FACE
 TYPES::DOMAIN_FACE_TYPE, 1002

DERIVATIVES_IN_LINE

TYPES::DOMAIN_LINE_TYPE, 1008

DETERMINANT_FULL_DP
 MATHS, 486

 MATHS::DETERMINANT, 872

DETERMINANT_FULL_INTG
 MATHS, 486

 MATHS::DETERMINANT, 872

DETERMINANT_FULL_SP
 MATHS, 487

 MATHS::DETERMINANT, 872

DGESV
 LAPACK::interface, 848

DIAG_ALL_SUBROUTINES
 BASE_ROUTINES, 171

DIAG_FILE_OPEN
 BASE_ROUTINES, 171

DIAG_FROM_SUBROUTINE
 BASE_ROUTINES, 171

DIAG_OR_TIMING
 BASE_ROUTINES, 171

DIAG_ROUTINE_LIST
 BASE_ROUTINES, 171

DIAGNOSTIC_OUTPUT_TYPE
 BASE_ROUTINES_OutputType, 19

DIAGNOSTICS
 BASE_ROUTINES, 171

 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 694

DIAGNOSTICS1
 BASE_ROUTINES, 171

DIAGNOSTICS2
 BASE_ROUTINES, 172

DIAGNOSTICS3
 BASE_ROUTINES, 172

DIAGNOSTICS4
 BASE_ROUTINES, 172

DIAGNOSTICS5
 BASE_ROUTINES, 172

DIAGNOSTICS_FILE_UNIT
 BASE_ROUTINES_FileUnits, 23

DIAGNOSTICS_LEVEL1
 BASE_ROUTINES, 172

DIAGNOSTICS_LEVEL2
 BASE_ROUTINES, 172

DIAGNOSTICS_LEVEL3
 BASE_ROUTINES, 172

DIAGNOSTICS_LEVEL4
 BASE_ROUTINES, 173

DIAGNOSTICS_LEVEL5
 BASE_ROUTINES, 173

DIAGNOSTICS_SET_OFF
 BASE_ROUTINES, 138

DIAGNOSTICS_SET_ON
 BASE_ROUTINES, 139

DIAGONAL_NUMBER_NON_ZEROS
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 976
 DIMENSION
 TYPES::FIELD_TYPE, 1109
 DIRECT_SOLVER
 TYPES::LINEAR_SOLVER_TYPE, 1127
 DIRECT_SOLVER_TYPE
 TYPES::LINEAR_DIRECT_SOLVER_-
 TYPE, 1123
 DISPLACEMENTS
 COMP_ENVIRONMENT::MPI_-
 COMPUTATIONAL_NODE_TYPE,
 739
 DISTRIBUTED_MATRIX
 TYPES::DISTRIBUTED_MATRIX_-
 CMISS_TYPE, 974
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 976
 DISTRIBUTED_VECTOR
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, 983
 TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, 985
 DIVERGENCE_TOLERANCE
 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1125
 DNRM2
 BLAS::interface, 714
 DOF_INDEX
 TYPES::DOMAIN_DOFS_TYPE, 995
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::MESH_NODE_TYPE, 1138
 DOF_LIST
 TYPES::FIELD_VARIABLE_TYPE, 1117
 DOF_TO_COLUMNS_MAPS
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1202
 DOF_TO_PARAM_MAP
 TYPES::FIELD_MAPPINGS_TYPE, 1095
 DOF_TO_ROWS_MAP
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1202
 DOF_TYPE
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1083
 DOFS
 TYPES::DOMAIN_MAPPINGS_TYPE, 1015
 TYPES::DOMAIN_TOPOLOGY_TYPE,
 1023
 TYPES::MESH_TOPOLOGY_TYPE, 1143
 DOMAIN
 TYPES::DECOMPOSITION_TYPE, 971
 TYPES::DOMAIN_DOFS_TYPE, 995
 TYPES::DOMAIN_ELEMENTS_TYPE, 999
 TYPES::DOMAIN_FACES_TYPE, 1004
 TYPES::DOMAIN_LINES_TYPE, 1010
 TYPES::DOMAIN_MAPPINGS_TYPE, 1015
 TYPES::DOMAIN_NODES_TYPE, 1020
 TYPES::DOMAIN_TOPOLOGY_TYPE,
 1023
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113
 DOMAIN_LOCAL_BOUNDARY
 DOMAIN_MAPPINGS_DomainType, 38
 DOMAIN_LOCAL_GHOST
 DOMAIN_MAPPINGS_DomainType, 38
 DOMAIN_LOCAL_INTERNAL
 DOMAIN_MAPPINGS_DomainType, 38
 DOMAIN_MAPPING
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 992
 TYPES::FIELD_MAPPINGS_TYPE, 1095
 TYPES::FIELD_VARIABLE_TYPE, 1117
 DOMAIN_MAPPINGS, 261
 DOMAIN_MAPPINGS_ADJACENT_-
 DOMAIN_FINALISE, 262
 DOMAIN_MAPPINGS_ADJACENT_-
 DOMAIN_INITIALISE, 262
 DOMAIN_MAPPINGS_LOCAL_FROM_-
 GLOBAL_CALCULATE, 262
 DOMAIN_MAPPINGS_MAPPING_-
 FINALISE, 263
 DOMAIN_MAPPINGS_MAPPING_-
 GLOBAL_FINALISE, 263
 DOMAIN_MAPPINGS_MAPPING_-
 GLOBAL_INITIALISE, 264
 DOMAIN_MAPPINGS_MAPPING_-
 INITIALISE, 264
 DOMAIN_MAPPINGS::DomainType, 38
 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_-
 FINALISE
 DOMAIN_MAPPINGS, 262
 DOMAIN_MAPPINGS_ADJACENT_DOMAIN_-
 INITIALISE
 DOMAIN_MAPPINGS, 262
 DOMAIN_MAPPINGS_DomainType
 DOMAIN_LOCAL_BOUNDARY, 38
 DOMAIN_LOCAL_GHOST, 38
 DOMAIN_LOCAL_INTERNAL, 38
 DOMAIN_MAPPINGS_LOCAL_FROM_-
 GLOBAL_CALCULATE
 DOMAIN_MAPPINGS, 262
 DOMAIN_MAPPINGS_MAPPING_FINALISE
 DOMAIN_MAPPINGS, 263
 DOMAIN_MAPPINGS_MAPPING_GLOBAL_-
 FINALISE
 DOMAIN_MAPPINGS, 263

DOMAIN_MAPPINGS_MAPPING_GLOBAL_-
INITIALISE
 DOMAIN_MAPPINGS, 264
DOMAIN_MAPPINGS_MAPPING_INITIALISE
 DOMAIN_MAPPINGS, 264
DOMAIN_MAPPINGS_NODES_FINALISE
 MESH_ROUTINES, 550
DOMAIN_MAPPINGS_NODES_INITIALISE
 MESH_ROUTINES, 551
DOMAIN_NUMBER
 TYPES::DOMAIN_ADJACENT_-
 DOMAIN_TYPE, 993
 TYPES::DOMAIN_GLOBAL_MAPPING_-
 TYPE, 1005
DOMAIN_TOPOLOGY_CALCULATE
 MESH_ROUTINES, 551
DOMAIN_TOPOLOGY_DOFS_FINALISE
 MESH_ROUTINES, 552
DOMAIN_TOPOLOGY_DOFS_INITIALISE
 MESH_ROUTINES, 552
DOMAIN_TOPOLOGY_ELEMENT_FINALISE
 MESH_ROUTINES, 553
DOMAIN_TOPOLOGY_ELEMENT_-
INITIALISE
 MESH_ROUTINES, 553
DOMAIN_TOPOLOGY_ELEMENTS_FINALISE
 MESH_ROUTINES, 553
DOMAIN_TOPOLOGY_ELEMENTS_-
INITIALISE
 MESH_ROUTINES, 554
DOMAIN_TOPOLOGY_FINALISE
 MESH_ROUTINES, 554
DOMAIN_TOPOLOGY_INITIALISE
 MESH_ROUTINES, 555
DOMAIN_TOPOLOGY_INITIALISE_FROM_-
MESH
 MESH_ROUTINES, 555
DOMAIN_TOPOLOGY_LINE_FINALISE
 MESH_ROUTINES, 556
DOMAIN_TOPOLOGY_LINE_INITIALISE
 MESH_ROUTINES, 556
DOMAIN_TOPOLOGY_LINES_FINALISE
 MESH_ROUTINES, 557
DOMAIN_TOPOLOGY_LINES_INITIALISE
 MESH_ROUTINES, 557
DOMAIN_TOPOLOGY_NODE_FINALISE
 MESH_ROUTINES, 558
DOMAIN_TOPOLOGY_NODE_INITIALISE
 MESH_ROUTINES, 558
DOMAIN_TOPOLOGY_NODES_FINALISE
 MESH_ROUTINES, 558
DOMAIN_TOPOLOGY_NODES_INITIALISE
 MESH_ROUTINES, 559
DOMAIN_TOPOLOGY_NODES_-
SURROUNDING_ELEMENTS_-
CALCULATE
 MESH_ROUTINES, 559
DOUBLE_COMPLEX_SIZE
 MACHINE_CONSTANTS, 480
DOUBLE_REAL_SIZE
 MACHINE_CONSTANTS, 480
DOUBLECOMPLEXTYPE
 binary_file_c.c, 1219
DOUBLETYPE
 binary_file_c.c, 1219
DP
 KINDS_RealKinds, 80
DP_FORMAT
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 698
DP_REAL_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
DPC
 KINDS_ComplexKinds, 83
DPC_REAL_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
DROT
 BLAS::interface, 714
DROTG
 BLAS::interface, 714
DSCAL
 BLAS::interface, 714
DTRSV
 BLAS::interface, 714
DX_DXI
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1088
DXI_DX
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1088
DXZ_DP
 COORDINATE_ROUTINES, 259
 COORDINATE_ROUTINES::DXZ, 755
DZX_DP
 COORDINATE_ROUTINES, 260
 COORDINATE_ROUTINES::DZX, 756
ECHO_FILE_UNIT
 BASE_ROUTINES_FileUnits, 23
ECHO_OUTPUT
 BASE_ROUTINES, 173
EDP_DP
 MATHS, 487
 MATHS::EDP, 873
EDP_SP

MATHS, 487
 MATHS::EDP, 873
EIGENPROBLEM_SOLVER
 TYPES::SOLVER_TYPE, 1197
EIGENVALUE_FULL_DP
 MATHS, 487
 MATHS::EIGENVALUE, 874
EIGENVALUE_FULL_SP
 MATHS, 487
 MATHS::EIGENVALUE, 874
EIGENVECTOR_FULL_DP
 MATHS, 487
 MATHS::EIGENVECTOR, 875
EIGENVECTOR_FULL_SP
 MATHS, 488
 MATHS::EIGENVECTOR, 875
ELASTICITY_ROUTINES, 265
ELECTROMECHANICS_ROUTINES, 266
ELEMENT_DERIVATIVES
 TYPES::DOMAIN_ELEMENT_TYPE, 997
ELEMENT_DOF2PARAM_MAP
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1083
ELEMENT_DOMAIN
 TYPES::DECOMPOSITION_TYPE, 971
ELEMENT_LINES
 TYPES::DECOMPOSITION_ELEMENT_-
 TYPE, 961
 TYPES::DECOMPOSITION_LINE_TYPE,
 965
ELEMENT_MATRIX
 TYPES::EQUATIONS_MATRIX_TYPE,
 1049
ELEMENT_NODES
 TYPES::DOMAIN_ELEMENT_TYPE, 997
ELEMENT_PARAM2DOF_MAP
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097
ELEMENT_PARAMETER_INDEX
 TYPES::BASIS_TYPE, 952
ELEMENT_VECTOR
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
ELEMENTAL_INFO_SET
 FIELD_IO_ROUTINES::FIELD_IO_-
 ELEMENTALL_INFO_SET, 761
ELEMENTS
 TYPES::DECOMPOSITION_ELEMENTS_-
 TYPE, 963
 TYPES::DECOMPOSITION_TOPOLOGY_-
 TYPE, 969
 TYPES::DOMAIN_ELEMENTS_TYPE, 999
 TYPES::DOMAIN_MAPPINGS_TYPE, 1015
TYPES::DOMAIN_TOPOLOGY_TYPE,
 1023
 TYPES::MESH_ELEMENTS_TYPE, 1136
 TYPES::MESH_TOPOLOGY_TYPE, 1143
ELEMENTS_FINISHED
 TYPES::MESH_ELEMENTS_TYPE, 1136
EMBEDDED_MESHES
 TYPES::MESH_TYPE, 1146
EMBEDDING_MESH
 TYPES::MESH_TYPE, 1146
ENDIAN_TYPE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
ENTERS
 BASE_ROUTINES, 139
EQ
 EQUATIONS_SET_ROUTINES, 280
EQUATIONS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1033
 TYPES::EQUATIONS_LINEAR_DATA_-
 TYPE, 1037
 TYPES::EQUATIONS_MAPPING_TYPE,
 1040
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
 TYPES::EQUATIONS_NONLINEAR_-
 DATA_TYPE, 1051
 TYPES::EQUATIONS_SET_TO_SOLVER_-
 MAP_TYPE, 1062
 TYPES::EQUATIONS_SET_TYPE, 1065
 TYPES::EQUATIONS_TIME_DATA_TYPE,
 1069
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SET_MAP_TYPE, 1182
EQUATIONS_COL_NUMBERS
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 MAP_TYPE, 1180
EQUATIONS_COL_SOLVER_COLS_MAP
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_TYPE, 1071
EQUATIONS_FINISHED
 TYPES::EQUATIONS_TYPE, 1077
EQUATIONS_INTERPOLATION_FINALISE
 EQUATIONS_SET_ROUTINES, 280
EQUATIONS_INTERPOLATION_INITIALISE
 EQUATIONS_SET_ROUTINES, 281
EQUATIONS_MAPPING
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
 TYPES::EQUATIONS_TYPE, 1077
EQUATIONS_MAPPING_FINISHED
 TYPES::EQUATIONS_MAPPING_TYPE,
 1040

EQUATIONS_MATRICES
 TYPES::EQUATIONS_MAPPING_TYPE,
 1040
 TYPES::EQUATIONS_MATRIX_TYPE,
 1049
 TYPES::EQUATIONS_TYPE, 1078
EQUATIONS_MATRICES_FINISHED
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
EQUATIONS_MATRIX
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_TYPE, 1071
EQUATIONS_MATRIX_NUMBER
 TYPES::ELEMENT_MATRIX_TYPE, 1028
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_TYPE, 1071
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_EM_TYPE, 1073
EQUATIONS_MATRIX_NUMBERS
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 MAP_TYPE, 1180
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1203
EQUATIONS_MATRIX_TO_VARIABLE_MAPS
 TYPES::EQUATIONS_MAPPING_TYPE,
 1040
EQUATIONS_ROW_NUMBER
 TYPES::SOLVER_ROW_TO_-
 EQUATIONS_SET_MAP_TYPE,
 1194
EQUATIONS_ROW_TO_SOLVER_ROWS_-
 MAPS
 TYPES::EQUATIONS_SET_TO_SOLVER_-
 MAP_TYPE, 1062
EQUATIONS_ROW_TO_SOURCE_DOF_MAP
 TYPES::SOURCE_EQUATIONS_-
 MATRICES_MAP_TYPE, 1199
EQUATIONS_ROW_TO_VARIABLES_MAPS
 TYPES::EQUATIONS_MAPPING_TYPE,
 1041
EQUATIONS_SET
 TYPES::EQUATIONS_SET_ANALYTIC_-
 TYPE, 1054
 TYPES::EQUATIONS_SET_DEPENDENT_-
 TYPE, 1055
 TYPES::EQUATIONS_SET_FIXED_-
 CONDITIONS_TYPE, 1056
 TYPES::EQUATIONS_SET_GEOMETRY_-
 TYPE, 1058
 TYPES::EQUATIONS_SET_MATERIALS_-
 TYPE, 1059
 TYPES::EQUATIONS_SET_SOURCE_-
 TYPE, 1061
 TYPES::EQUATIONS_TYPE, 1078
 TYPES::SOLVER_ROW_TO_-
 EQUATIONS_SET_MAP_TYPE,
 1194
EQUATIONS_SET_ADDED_INDEX
 TYPES::SOLUTION_TYPE, 1178
EQUATIONS_SET_ADVECTION_DIFFUSION_-
 EQUATION_TYPE
 EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_ANALYTIC_CREATE_-
 FINISH
 EQUATIONS_SET_ROUTINES, 281
EQUATIONS_SET_ANALYTIC_CREATE_-
 START
 EQUATIONS_SET_ROUTINES, 282
EQUATIONS_SET_ANALYTIC_DESTROY
 EQUATIONS_SET_ROUTINES, 282
EQUATIONS_SET_ANALYTIC_FINALISE
 EQUATIONS_SET_ROUTINES, 282
EQUATIONS_SET_ANALYTIC_INITIALISE
 EQUATIONS_SET_ROUTINES, 283
EQUATIONS_SET_ASSEMBLE
 EQUATIONS_SET_ROUTINES, 283
EQUATIONS_SET_ASSEMBLE_LINEAR_-
 STATIC_FEM
 EQUATIONS_SET_ROUTINES, 284
EQUATIONS_SET_BACKSUBSTITUTE
 EQUATIONS_SET_ROUTINES, 284
EQUATIONS_SET_BEM SOLUTION_-
 METHOD
 EQUATIONS_SET_CONSTANTS_-
 SolutionMethods, 52
EQUATIONS_SET_BIHAMONIC_-
 EQUATION_TYPE
 EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_CLASSICAL_FIELD_CLASS
 EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_CONSTANTS, 267
 EQUATIONS_SET_ADVECTION_-
 DIFFUSION_EQUATION_TYPE,
 270
 EQUATIONS_SET_BIHAMONIC_-
 EQUATION_TYPE, 270
 EQUATIONS_SET_CLASSICAL_FIELD_-
 CLASS, 270
 EQUATIONS_SET_DIFFUSION_-
 EQUATION_TYPE, 270
 EQUATIONS_SET_ELASTICITY_CLASS,
 270
 EQUATIONS_SET_-
 ELECTROMAGNETICS_CLASS,
 270

EQUATIONS_SET_ELECTROSTATIC_-
 TYPE, 271
 EQUATIONS_SETFINITE_ELASTICITY_-
 TYPE, 271
 EQUATIONS_SETFITTING_CLASS, 271
 EQUATIONS_SETFLUID_MECHANICS_-
 CLASS, 271
 EQUATIONS_SETGENERALISED_-
 LAPLACE_SUBTYPE, 271
 EQUATIONS_SETHELMHOLTZ_-
 EQUATION_TYPE, 271
 EQUATIONS_SETLAPLACE_-
 EQUATION_TYPE, 271
 EQUATIONS_SETLINEAR_ELASTIC_-
 MODAL_TYPE, 272
 EQUATIONS_SETLINEAR_-
 ELASTICITY_TYPE, 272
 EQUATIONS_SETMAGNETOSTATIC_-
 TYPE, 272
 EQUATIONS_SETMAXWELLS_-
 EQUATIONS_TYPE, 272
 EQUATIONS_SETMODAL_CLASS, 272
 EQUATIONS_SETNAVIER_STOKES_-
 FLUID_TYPE, 272
 EQUATIONS_SETNO_CLASS, 272
 EQUATIONS_SETNO_SUBTYPE, 273
 EQUATIONS_SETNO_TYPE, 273
 EQUATIONS_SETOPTIMISATION_-
 CLASS, 273
 EQUATIONS_SETPOISSON_-
 EQUATION_TYPE, 273
 EQUATIONS_SETREACTION_-
 DIFFUSION_EQUATION_TYPE,
 273
 EQUATIONS_SETSTANDARD_-
 LAPLACE_SUBTYPE, 273
 EQUATIONS_SETSTOKES_FLUID_-
 TYPE, 273
 EQUATIONS_SETWAVE_EQUATION_-
 TYPE, 274
 EQUATIONS_SET_-
 CONSTANTS::FixedConditions, 46
 EQUATIONS_SET_-
 CONSTANTS::LinearityTypes, 48
 EQUATIONS_SETCONSTANTS::OutputTypes,
 55
 EQUATIONS_SET_-
 CONSTANTS::SetupActionTypes,
 44
 EQUATIONS_SETCONSTANTS::SetupTypes,
 40
 EQUATIONS_SET_-
 CONSTANTS::SolutionMethods, 52
 EQUATIONS_SETCONSTANTS::SparsityTypes,
 57
 EQUATIONS_SET_-
 CONSTANTS::TimeDepedenceTypes,
 50
 EQUATIONS_SETCONSTANTS_-
 FixedConditions
 EQUATIONS_SETFIXED_BOUNDARY_-
 CONDITION, 46
 EQUATIONS_SETMIXED_BOUNDARY_-
 CONDITION, 46
 EQUATIONS_SETNOT_FIXED, 47
 EQUATIONS_SETCONSTANTS_LinearityTypes
 EQUATIONS_SET_LINEAR, 48
 EQUATIONS_SET_NONLINEAR, 48
 EQUATIONS_SET_NONLINEAR_BCS, 49
 NUMBER_OF_EQUATIONS_SET_-
 LINEARITY_TYPES, 49
 EQUATIONS_SETCONSTANTS_OutputTypes
 EQUATIONS_SET_ELEMENT_MATRIX_-
 OUTPUT, 55
 EQUATIONS_SET_MATRIX_OUTPUT, 55
 EQUATIONS_SET_NO_OUTPUT, 56
 EQUATIONS_SET_TIMING_OUTPUT, 56
 EQUATIONS_SETCONSTANTS_-
 SetupActionTypes
 EQUATIONS_SETSETUP_FINISH_-
 ACTION, 44
 EQUATIONS_SETSETUP_START_-
 ACTION, 44
 EQUATIONS_SETCONSTANTS_SetupTypes
 EQUATIONS_SET_SETUP_ANALYTIC_-
 TYPE, 41
 EQUATIONS_SET_SETUP_DEPENDENT_-
 TYPE, 41
 EQUATIONS_SET_SETUP_EQUATIONS_-
 TYPE, 41
 EQUATIONS_SET_SETUP_FINAL_TYPE,
 42
 EQUATIONS_SET_SETUP_FIXED_-
 CONDITIONS_TYPE, 42
 EQUATIONS_SET_SETUP_GEOMETRY_-
 TYPE, 42
 EQUATIONS_SET_SETUP_INITIAL_-
 TYPE, 42
 EQUATIONS_SET_SETUP_MATERIALS_-
 TYPE, 43
 EQUATIONS_SET_SETUP_SOURCE_-
 MATERIALS_TYPE, 43
 EQUATIONS_SET_SETUP_SOURCE_-
 TYPE, 43
 EQUATIONS_SETCONSTANTS_-
 SolutionMethods
 EQUATIONS_SETBEM SOLUTION_-
 METHOD, 52

EQUATIONS_SET_FD SOLUTION_-
METHOD, 53
EQUATIONS_SET_FEM SOLUTION_-
METHOD, 53
EQUATIONS_SET_FV SOLUTION_-
METHOD, 53
EQUATIONS_SET_GFEM SOLUTION_-
METHOD, 53
EQUATIONS_SET_GFV SOLUTION_-
METHOD, 54
NUMBER_OF_EQUATIONS_SET_-
SOLUTION_METHODS, 54
EQUATIONS_SET_CONSTANTS_SparsityTypes
EQUATIONS_SET_FULL_MATRICES, 57
EQUATIONS_SET_SPARSE_MATRICES,
57
EQUATIONS_SET_CONSTANTS_-
TimeDepedenceTypes
EQUATIONS_SET_DYNAMIC, 50
EQUATIONS_SET_QUASISTATIC, 50
EQUATIONS_SET_STATIC, 51
NUMBER_OF_EQUATIONS_SET_TIME_-
TYPES, 51
EQUATIONS_SET_CREATE_FINISH
EQUATIONS_SET_ROUTINES, 285
EQUATIONS_SET_CREATE_START
EQUATIONS_SET_ROUTINES, 285
EQUATIONS_SET_DEPENDENT_
EQUATIONS_SET_ROUTINES, 286
EQUATIONS_SET_DEPENDENT_CREATE_-
FINISH
EQUATIONS_SET_ROUTINES, 286
EQUATIONS_SET_DEPENDENT_CREATE_-
START
EQUATIONS_SET_ROUTINES, 287
EQUATIONS_SET_DEPENDENT_-
DEPENDENT_FIELD_GET
EQUATIONS_SET_ROUTINES, 287
EQUATIONS_SET_DEPENDENT_DESTROY
EQUATIONS_SET_ROUTINES, 287
EQUATIONS_SET_DEPENDENT_FINALISE
EQUATIONS_SET_ROUTINES, 288
EQUATIONS_SET_DEPENDENT_INITIALISE
EQUATIONS_SET_ROUTINES, 288
EQUATIONS_SET_DEPENDENT_SCALING_-
SET
EQUATIONS_SET_ROUTINES, 289
EQUATIONS_SET_DESTROY
EQUATIONS_SET_ROUTINES, 289
EQUATIONS_SET_DIFFUSION_EQUATION_-
TYPE
EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_DYNAMIC
EQUATIONS_SET_CONSTANTS -,
TimeDepedenceTypes, 50
EQUATIONS_SET_ELASTICITY_CLASS
EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_ELECTROMAGNETICS_-
CLASS
EQUATIONS_SET_CONSTANTS, 270
EQUATIONS_SET_ELECTROSTATIC_TYPE
EQUATIONS_SET_CONSTANTS, 271
EQUATIONS_SET_ELEMENT_MATRIX_-
OUTPUT
EQUATIONS_SET_CONSTANTS -,
OutputTypes, 55
EQUATIONS_SET_EQUATIONS_CREATE_-
FINISH
EQUATIONS_SET_ROUTINES, 289
EQUATIONS_SET_EQUATIONS_CREATE_-
START
EQUATIONS_SET_ROUTINES, 290
EQUATIONS_SET_EQUATIONS_FINALISE
EQUATIONS_SET_ROUTINES, 290
EQUATIONS_SET_EQUATIONS_INITIALISE
EQUATIONS_SET_ROUTINES, 290
EQUATIONS_SET_EQUATIONS_LINEAR_-
DATA_FINALISE
EQUATIONS_SET_ROUTINES, 291
EQUATIONS_SET_EQUATIONS_LINEAR_-
DATA_INITIALISE
EQUATIONS_SET_ROUTINES, 291
EQUATIONS_SET_EQUATIONS -,
NONLINEAR_DATA_FINALISE
EQUATIONS_SET_ROUTINES, 292
EQUATIONS_SET_EQUATIONS -,
NONLINEAR_DATA_INITIALISE
EQUATIONS_SET_ROUTINES, 292
EQUATIONS_SET_EQUATIONS_OUTPUT_-
TYPE_SET
EQUATIONS_SET_ROUTINES, 292
EQUATIONS_SET_EQUATIONS_SPARSITY_-
TYPE_SET
EQUATIONS_SET_ROUTINES, 293
EQUATIONS_SET_EQUATIONS_TIME_DATA_-
FINALISE
EQUATIONS_SET_ROUTINES, 293
EQUATIONS_SET_EQUATIONS_TIME_DATA_-
INITIALISE
EQUATIONS_SET_ROUTINES, 294
EQUATIONS_SET_FD SOLUTION_METHOD
EQUATIONS_SET_CONSTANTS -,
SolutionMethods, 53
EQUATIONS_SET_FEM SOLUTION_METHOD
EQUATIONS_SET_CONSTANTS -,
SolutionMethods, 53
EQUATIONS_SET_FINALISE

EQUATIONS_SET_ROUTINES, 294
 EQUATIONS_SET_FINISHED
 TYPES::EQUATIONS_SET_TYPE, 1065
 EQUATIONS_SETFINITE_ELASTICITY_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SETFINITE_ELEMENT_-
 CALCULATE
 EQUATIONS_SET_ROUTINES, 294
 EQUATIONS_SETFITTING_CLASS
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SET_FIXED_BOUNDARY_-
 CONDITION
 EQUATIONS_SET_CONSTANTS_-
 FixedConditions, 46
 EQUATIONS_SET_FIXED_CONDITIONS_-
 APPLY
 EQUATIONS_SET_ROUTINES, 295
 EQUATIONS_SET_FIXED_CONDITIONS_-
 CREATE_FINISH
 EQUATIONS_SET_ROUTINES, 295
 EQUATIONS_SET_FIXED_CONDITIONS_-
 CREATE_START
 EQUATIONS_SET_ROUTINES, 296
 EQUATIONS_SET_FIXED_CONDITIONS_-
 DESTROY
 EQUATIONS_SET_ROUTINES, 296
 EQUATIONS_SET_FIXED_CONDITIONS_-
 FINALISE
 EQUATIONS_SET_ROUTINES, 297
 EQUATIONS_SET_FIXED_CONDITIONS_-
 INITIALISE
 EQUATIONS_SET_ROUTINES, 297
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOF1
 EQUATIONS_SET_ROUTINES, 297
 EQUATIONS_SET_-
 ROUTINES::EQUATIONS_SET_-
 FIXED_CONDITIONS_SET_DOF,
 757
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOFS
 EQUATIONS_SET_ROUTINES, 298
 EQUATIONS_SET_-
 ROUTINES::EQUATIONS_SET_-
 FIXED_CONDITIONS_SET_DOF,
 757
 EQUATIONS_SET_FLUID_MECHANICS_-
 CLASS
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SET_FULL_MATRICES
 EQUATIONS_SET_CONSTANTS_-
 SparsityTypes, 57
 EQUATIONS_SET_FV SOLUTION_METHOD
 EQUATIONS_SET_CONSTANTS_-
 SolutionMethods, 53
 EQUATIONS_SET_GENERALISED_-
 LAPLACE_SUBTYPE
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SET_GEOMETRY_FINALISE
 EQUATIONS_SET_ROUTINES, 299
 EQUATIONS_SET_GEOMETRY_INITIALISE
 EQUATIONS_SET_ROUTINES, 299
 EQUATIONS_SET_GFEM SOLUTION_-
 METHOD
 EQUATIONS_SET_CONSTANTS_-
 SolutionMethods, 53
 EQUATIONS_SET_GFV SOLUTION_METHOD
 EQUATIONS_SET_CONSTANTS_-
 SolutionMethods, 54
 EQUATIONS_SET_HELMHOLTZ_EQUATION_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SET_INDEX
 TYPES::EQUATIONS_SET_TO_SOLVER_-
 MAP_TYPE, 1063
 EQUATIONS_SET_INDICES
 TYPES::SOLVER_COL_TO_VARIABLE_-
 MAP_TYPE, 1186
 EQUATIONS_SET_INITIALISE
 EQUATIONS_SET_ROUTINES, 299
 EQUATIONS_SET_LAPLACE_EQUATION_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 271
 EQUATIONS_SET_LINEAR
 EQUATIONS_SET_CONSTANTS_-
 LinearityTypes, 48
 EQUATIONS_SET_LINEAR_ELASTIC_-
 MODAL_TYPE
 EQUATIONS_SET_CONSTANTS, 272
 EQUATIONS_SET_LINEAR_ELASTICITY_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 272
 EQUATIONS_SET_MAGNETOSTATIC_TYPE
 EQUATIONS_SET_CONSTANTS, 272
 EQUATIONS_SET_MATERIALS_-
 COMPONENT_INTERPOLATION_-
 SET
 EQUATIONS_SET_ROUTINES, 300
 EQUATIONS_SET_MATERIALS_-
 COMPONENT_MESH_-
 COMPONENT_SET
 EQUATIONS_SET_ROUTINES, 300
 EQUATIONS_SET_MATERIALS_CREATE_-
 FINISH
 EQUATIONS_SET_ROUTINES, 301
 EQUATIONS_SET_MATERIALS_CREATE_-
 START

EQUATIONS_SET_ROUTINES, 301
EQUATIONS_SET_MATERIALS_DESTROY
 EQUATIONS_SET_ROUTINES, 301
EQUATIONS_SET_MATERIALS_FINALISE
 EQUATIONS_SET_ROUTINES, 302
EQUATIONS_SET_MATERIALS_INITIALISE
 EQUATIONS_SET_ROUTINES, 302
EQUATIONS_SET_MATERIALS_MATERIAL_-
 FIELD_GET
 EQUATIONS_SET_ROUTINES, 303
EQUATIONS_SET_MATERIALS_SCALING_-
 SET
 EQUATIONS_SET_ROUTINES, 303
EQUATIONS_SET_MATRIX_OUTPUT
 EQUATIONS_SET_CONSTANTS_-
 OutputTypes, 55
EQUATIONS_SET_MAXWELLS_-
 EQUATIONS_TYPE
 EQUATIONS_SET_CONSTANTS, 272
EQUATIONS_SET_MIXED_BOUNDARY_-
 CONDITION
 EQUATIONS_SET_CONSTANTS_-
 FixedConditions, 46
EQUATIONS_SET_MODAL_CLASS
 EQUATIONS_SET_CONSTANTS, 272
EQUATIONS_SET_NAVIER_STOKES_FLUID_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 272
EQUATIONS_SET_NO_CLASS
 EQUATIONS_SET_CONSTANTS, 272
EQUATIONS_SET_NO_OUTPUT
 EQUATIONS_SET_CONSTANTS_-
 OutputTypes, 56
EQUATIONS_SET_NO_SUBTYPE
 EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_NO_TYPE
 EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_NONLINEAR
 EQUATIONS_SET_CONSTANTS_-
 LinearityTypes, 48
EQUATIONS_SET_NONLINEAR_BCS
 EQUATIONS_SET_CONSTANTS_-
 LinearityTypes, 49
EQUATIONS_SET_NOT_FIXED
 EQUATIONS_SET_CONSTANTS_-
 FixedConditions, 47
EQUATIONS_SET_OPTIMISATION_CLASS
 EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_POISSON_EQUATION_-
 TYPE
 EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_QUASISTATIC
 EQUATIONS_SET_CONSTANTS_-
 TimeDepedenceTypes, 50

EQUATIONS_SETREACTION_DIFFUSION_-
 EQUATION_TYPE
 EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_ROUTINES, 275
 EQ, 280
 EQUATIONS_INTERPOLATION_-
 FINALISE, 280
 EQUATIONS_INTERPOLATION_-
 INITIALISE, 281
EQUATIONS_SET_ANALYTIC_CREATE_-
 FINISH, 281
EQUATIONS_SET_ANALYTIC_CREATE_-
 START, 282
EQUATIONS_SET_ANALYTIC_DESTROY,
 282
EQUATIONS_SET_ANALYTIC_FINALISE,
 282
EQUATIONS_SET_ANALYTIC_-
 INITIALISE, 283
EQUATIONS_SET_ASSEMBLE, 283
EQUATIONS_SET_ASSEMBLE_LINEAR_-
 STATIC_FEM, 284
EQUATIONS_SET_BACKSUBSTITUTE,
 284
EQUATIONS_SET_CREATE_FINISH, 285
EQUATIONS_SET_CREATE_START, 285
EQUATIONS_SET_DEPENDENT_, 286
EQUATIONS_SET_DEPENDENT_-
 CREATE_FINISH, 286
EQUATIONS_SET_DEPENDENT_-
 CREATE_START, 287
EQUATIONS_SET_DEPENDENT_-
 DEPENDENT_FIELD_GET, 287
EQUATIONS_SET_DEPENDENT_-
 DESTROY, 287
EQUATIONS_SET_DEPENDENT_-
 FINALISE, 288
EQUATIONS_SET_DEPENDENT_-
 INITIALISE, 288
EQUATIONS_SET_DEPENDENT_-
 SCALING_SET, 289
EQUATIONS_SET_DESTROY, 289
EQUATIONS_SET_EQUATIONS_-
 CREATE_FINISH, 289
EQUATIONS_SET_EQUATIONS_-
 CREATE_START, 290
EQUATIONS_SET_EQUATIONS_-
 FINALISE, 290
EQUATIONS_SET_EQUATIONS_-
 INITIALISE, 290
EQUATIONS_SET_EQUATIONS_-
 LINEAR_DATA_FINALISE, 291
EQUATIONS_SET_EQUATIONS_-
 LINEAR_DATA_INITIALISE, 291

EQUATIONS_SET_EQUATIONS_-
 NONLINEAR_DATA_FINALISE,
 292
 EQUATIONS_SET_EQUATIONS_-
 NONLINEAR_DATA_INITIALISE,
 292
 EQUATIONS_SET_EQUATIONS_-
 OUTPUT_TYPE_SET, 292
 EQUATIONS_SET_EQUATIONS_-
 SPARSITY_TYPE_SET, 293
 EQUATIONS_SET_EQUATIONS_TIME_-
 DATA_FINALISE, 293
 EQUATIONS_SET_EQUATIONS_TIME_-
 DATA_INITIALISE, 294
 EQUATIONS_SET_FINALISE, 294
 EQUATIONS_SETFINITE_ELEMENT_-
 CALCULATE, 294
 EQUATIONS_SET_FIXED_CONDITIONS_-
 APPLY, 295
 EQUATIONS_SET_FIXED_CONDITIONS_-
 CREATE_FINISH, 295
 EQUATIONS_SET_FIXED_CONDITIONS_-
 CREATE_START, 296
 EQUATIONS_SET_FIXED_CONDITIONS_-
 DESTROY, 296
 EQUATIONS_SET_FIXED_CONDITIONS_-
 FINALISE, 297
 EQUATIONS_SET_FIXED_CONDITIONS_-
 INITIALISE, 297
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOF1, 297
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOFS, 298
 EQUATIONS_SET_GEOMETRY_-
 FINALISE, 299
 EQUATIONS_SET_GEOMETRY_-
 INITIALISE, 299
 EQUATIONS_SET_INITIALISE, 299
 EQUATIONS_SET_MATERIALS_-
 COMPONENT_INTERPOLATION_-
 SET, 300
 EQUATIONS_SET_MATERIALS_-
 COMPONENT_MESH_-
 COMPONENT_SET, 300
 EQUATIONS_SET_MATERIALS_-
 CREATE_FINISH, 301
 EQUATIONS_SET_MATERIALS_-
 CREATE_START, 301
 EQUATIONS_SET_MATERIALS_-
 DESTROY, 301
 EQUATIONS_SET_MATERIALS_-
 FINALISE, 302
 EQUATIONS_SET_MATERIALS_-
 INITIALISE, 302

 EQUATIONS_SET_MATERIALS_-
 MATERIAL_FIELD_GET, 303
 EQUATIONS_SET_MATERIALS_-
 SCALING_SET, 303
 EQUATIONS_SET_SETUP, 303
 EQUATIONS_SET_SOURCE_CREATE_-
 FINISH, 304
 EQUATIONS_SET_SOURCE_CREATE_-
 START, 304
 EQUATIONS_SET_SOURCE_DESTROY,
 305
 EQUATIONS_SET_SOURCE_FINALISE,
 305
 EQUATIONS_SET_SOURCE_INITIALISE,
 306
 EQUATIONS_SET_SOURCE_SCALING_-
 SET, 306
 EQUATIONS_SET_SPECIFICAT, 306
 EQUATIONS_SET_USER_NUMBER_-
 FIND, 307
 EQUATIONS_SETS_FINALISE, 307
 EQUATIONS_SETS_INITIALISE, 308
 EQUATIONS_SET_ROUTINES::EQUATIONS_-
 SET_FIXED_CONDITIONS_SET_-
 DOF, 757
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOF1, 757
 EQUATIONS_SET_FIXED_CONDITIONS_-
 SET_DOFS, 757
 EQUATIONS_SET_ROUTINES::EQUATIONS_-
 SET_SPECIFICATION_SET, 759
 EQUATIONS_SET_SPECIFICATION_SET_-
 NUMBER, 759
 EQUATIONS_SET_SPECIFICATION_SET_-
 PTR, 759
 EQUATIONS_SET_SETUP
 EQUATIONS_SET_ROUTINES, 303
 EQUATIONS_SET_SETUP_ANALYTIC_TYPE
 EQUATIONS_SET_CONSTANTS_-
 SetupTypes, 41
 EQUATIONS_SET_SETUP_DEPENDENT_-
 TYPE
 EQUATIONS_SET_CONSTANTS_-
 SetupTypes, 41
 EQUATIONS_SET_SETUP_EQUATIONS_TYPE
 EQUATIONS_SET_CONSTANTS_-
 SetupTypes, 41
 EQUATIONS_SET_SETUP_FINAL_TYPE
 EQUATIONS_SET_CONSTANTS_-
 SetupTypes, 42
 EQUATIONS_SET_SETUP_FINISH_ACTION
 EQUATIONS_SET_CONSTANTS_-
 SetupActionTypes, 44

EQUATIONS_SET_SETUP_FIXED_-
CONDITIONS_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 42
EQUATIONS_SET_SETUP_GEOMETRY_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 42
EQUATIONS_SET_SETUP_INITIAL_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 42
EQUATIONS_SET_SETUP_MATERIALS_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 43
EQUATIONS_SET_SETUP_SOURCE_-
MATERIALS_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 43
EQUATIONS_SET_SETUP_SOURCE_TYPE
EQUATIONS_SET_CONSTANTS_-
SetupTypes, 43
EQUATIONS_SET_SETUP_START_ACTION
EQUATIONS_SET_CONSTANTS_-
SetupActionTypes, 44
EQUATIONS_SET_SOURCE_CREATE_FINISH
EQUATIONS_SET_ROUTINES, 304
EQUATIONS_SET_SOURCE_CREATE_START
EQUATIONS_SET_ROUTINES, 304
EQUATIONS_SET_SOURCE_DESTROY
EQUATIONS_SET_ROUTINES, 305
EQUATIONS_SET_SOURCE_FINALISE
EQUATIONS_SET_ROUTINES, 305
EQUATIONS_SET_SOURCE_INITIALISE
EQUATIONS_SET_ROUTINES, 306
EQUATIONS_SET_SOURCE_SCALING_SET
EQUATIONS_SET_ROUTINES, 306
EQUATIONS_SET_SPARSE_MATRICES
EQUATIONS_SET_CONSTANTS_-
SparsityTypes, 57
EQUATIONS_SET_SPECIFICAT
EQUATIONS_SET_ROUTINES, 306
EQUATIONS_SET_SPECIFICATION_SET_-
NUMBER
EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_-
SPECIFICATION_SET, 759
EQUATIONS_SET_SPECIFICATION_SET_PTR
EQUATIONS_SET_-
ROUTINES::EQUATIONS_SET_-
SPECIFICATION_SET, 759
EQUATIONS_SET_STANDARD_LAPLACE_-
SUBTYPE
EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_STATIC
EQUATIONS_SET_CONSTANTS_-
TimeDepedenceTypes, 51
EQUATIONS_SET_STOKES_FLUID_TYPE
EQUATIONS_SET_CONSTANTS, 273
EQUATIONS_SET_TIMING_OUTPUT
EQUATIONS_SET_CONSTANTS_-
OutputTypes, 56
EQUATIONS_SET_TO_ADD
TYPES::SOLUTION_TYPE, 1178
EQUATIONS_SET_TO_SOLVER_MAP
TYPES::SOLUTION_MAPPING_TYPE,
1175
EQUATIONS_SET_USER_NUMBER_FIND
EQUATIONS_SET_ROUTINES, 307
EQUATIONS_SET_WAVE_EQUATION_TYPE
EQUATIONS_SET_CONSTANTS, 274
EQUATIONS_SETS
TYPES::EQUATIONS_SET_TYPE, 1065
TYPES::EQUATIONS_SETS_TYPE, 1068
TYPES::REGION_TYPE, 1171
TYPES::SOLUTION_MAPPING_TYPE,
1175
EQUATIONS_SETS_FINALISE
EQUATIONS_SET_ROUTINES, 307
EQUATIONS_SETS_INITIALISE
EQUATIONS_SET_ROUTINES, 308
EQUATIONS_TO_SOLVER_MATRIX_MAPS
TYPES::EQUATIONS_TO_SOLVER_-
MATRIX_MAPS_EM_TYPE, 1073
TYPES::EQUATIONS_TO_SOLVER_-
MATRIX_MAPS_SM_TYPE, 1075
EQUATIONS_TO_SOLVER_MATRIX_MAPS_-
EM
TYPES::EQUATIONS_SET_TO_SOLVER_-
MAP_TYPE, 1063
EQUATIONS_TO_SOLVER_MATRIX_MAPS_-
SM
TYPES::EQUATIONS_SET_TO_SOLVER_-
MAP_TYPE, 1063
ERROR_OUTPUT_TYPE
BASE_ROUTINES_OutputType, 20
ERROR_SEPARATOR_CONSTANT
MACHINE_CONSTANTS, 481
ERRORS
BASE_ROUTINES, 148
EXCLUSIVE_CPU_TIME
BASE_ROUTINES::ROUTINE_STACK_-
ITEM_TYPE, 694
EXCLUSIVE_SYSTEM_TIME
BASE_ROUTINES::ROUTINE_STACK_-
ITEM_TYPE, 695
EXITS
BASE_ROUTINES, 158
extract_CH

ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::extract, 822
 extract_VS
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::extract, 822

 F2CSTRING
 F90C, 310
 F90C, 309
 C2FSTRING, 309
 CSTRINGLENGTH, 310
 F2CSTRING, 310
 FSTRINGLENGTH, 310
 F90C::interface, 760
 CSTRINGLEN, 760
 PACKCHARACTERS, 760
 UNPACKCHARACTERS, 760
 f90c_c.c
 CStringLen, 1272
 integer, 1272
 logical, 1272
 PackCharacters, 1272
 UnPackCharacters, 1272
 FACE_BASES
 TYPES::BASIS_TYPE, 952
 FACES
 TYPES::DOMAIN_FACES_TYPE, 1004
 TYPES::DOMAIN_TOPOLOGY_TYPE,
 1024
 FAMILY_NUMBER
 TYPES::BASIS_TYPE, 952
 FEM_ELEMENT_MATRICES_FINALISE
 FINITE_ELEMENT_ROUTINES, 352
 FEM_ELEMENT_MATRICES_INITIALISE
 FINITE_ELEMENT_ROUTINES, 352
 FI
 FIELD_ROUTINES, 336
 FIBRE_FIELD
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 TYPES::EQUATIONS_SET_GEOMETRY_-
 TYPE, 1058
 FIBRE_INTERP_PARAMETERS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 FIBRE_INTERP_POINT
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 FIBRE_INTERP_POINT_METRICS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 FIE
 FIELD_IO_ROUTINES, 315
 FIELD
 TYPES::FIELD_INTERPOLATION_-
 PARAMETERS_TYPE, 1093
 TYPES::FIELD_PARAMETER_SETS_-
 TYPE, 1102
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113
 TYPES::FIELD_VARIABLE_TYPE, 1117
 FIELD_ARC_LENGTH_SCALING
 FIELD_ROUTINES_ScalingTypes, 71
 FIELD_ARITHMETIC_MEAN_SCALING
 FIELD_ROUTINES_ScalingTypes, 71
 FIELD_BOUNDARY_CONDITIONS_SET_-
 TYPE
 FIELD_ROUTINES_ParameterSetTypes, 69
 FIELD_COMPONENT_INTER
 FIELD_ROUTINES, 337
 FIELD_COMPONENT_INTERPOLATION_-
 SET_NUMBER
 FIELD_ROUTINES::FIELD_-
 COMPONENT_INTERPOLATION_-
 SET, 768
 FIELD_COMPONENT_INTERPOLATION_-
 SET_PTR
 FIELD_ROUTINES::FIELD_-
 COMPONENT_INTERPOLATION_-
 SET, 768
 FIELD_COMPONENT_MESH_COM
 FIELD_ROUTINES, 337
 FIELD_COMPONENT_MESH_COMPONENT_-
 SET_NUMBER
 FIELD_ROUTINES::FIELD_-
 COMPONENT_MESH_-
 COMPONENT_SET, 769
 FIELD_COMPONENT_MESH_COMPONENT_-
 SET_PTR
 FIELD_ROUTINES::FIELD_-
 COMPONENT_MESH_-
 COMPONENT_SET, 769
 FIELD_CONSTANT_DOF_TYPE
 FIELD_ROUTINES_DofTypes, 67
 FIELD_CONSTANT_INTERPOLATION
 FIELD_ROUTINES_InterpolationTypes, 62
 FIELD_CREATE_FINISH
 FIELD_ROUTINES, 338
 FIELD_CREATE_VALUES_CACHE_FINALISE
 FIELD_ROUTINES, 338
 FIELD_CREATE_VALUES_CACHE_-
 INITIALISE
 FIELD_ROUTINES, 339
 FIELD_DEPENDENT_TYPE
 FIELD_ROUTINES_DependentTypes, 58
 FIELD_DEPENDENT_TYPE_SET_NUMBER
 FIELD_ROUTINES, 339

FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_SET, 770
FIELD_DEPENDENT_TYPE_SET_PTR
FIELD_ROUTINES, 340
FIELD_ROUTINES::FIELD_-
DEPENDENT_TYPE_SET, 770
FIELD_DESTROY
FIELD_ROUTINES, 340
FIELD_DIMENSION_SET_NUMBER
FIELD_ROUTINES, 341
FIELD_ROUTINES::FIELD_DIMENSION_-
SET, 771
FIELD_DIMENSION_SET_PTR
FIELD_ROUTINES, 341
FIELD_ROUTINES::FIELD_DIMENSION_-
SET, 771
FIELD_ELEMENT_BASED_INTERPOLATION
FIELD_ROUTINES_InterpolationTypes, 62
FIELD_ELEMENT_DOF_TYPE
FIELD_ROUTINES_DofTypes, 67
FIELD_FIBRE_TYPE
FIELD_ROUTINES_FieldTypes, 60
FIELD_FINISHED
TYPES::FIELD_TYPE, 1109
FIELD_GENERAL_TYPE
FIELD_ROUTINES_FieldTypes, 60
FIELD_GEOMETRIC_FIELD_SET_NUMBER
FIELD_ROUTINES::FIELD_-
GEOMETRIC_FIELD_SET, 772
FIELD_GEOMETRIC_FIELD_SET_PTR
FIELD_ROUTINES::FIELD_-
GEOMETRIC_FIELD_SET, 772
FIELD_GEOMETRIC_TYPE
FIELD_ROUTINES_FieldTypes, 61
FIELD_HARMONIC_MEAN_SCALING
FIELD_ROUTINES_ScalingTypes, 72
FIELD_INDEPENDENT_TYPE
FIELD_ROUTINES_DependentTypes, 58
FIELD_INITIAL_CONDITIONS_SET_TYPE
FIELD_ROUTINES_ParameterSetTypes, 69
FIELD_INTERPOLATE_GAUSS
FIELD_ROUTINES, 342
FIELD_INTERPOLATE_XI
FIELD_ROUTINES, 342
FIELD_INTERPOLATED_POINT_FINALISE
FIELD_ROUTINES, 343
FIELD_INTERPOLATED_POINT_INITIALISE
FIELD_ROUTINES, 343
FIELD_INTERPOLATED_POINT_METRICS_-
CALCULATE
FIELD_ROUTINES, 344
FIELD_INTERPOLATED_POINT_METRICS_-
FINALISE
FIELD_ROUTINES, 344
FIELD_INTERPOLATED_POINT_METRICS_-
INITIALISE
FIELD_ROUTINES, 345
FIELD_INTERPOLATION_PARAMETERS_-
ELEMENT_GET
FIELD_ROUTINES, 345
FIELD_INTERPOLATION_PARAMETERS_-
FINALISE
FIELD_ROUTINES, 346
FIELD_INTERPOLATION_PARAMETERS_-
INITIALISE
FIELD_ROUTINES, 347
FIELD_INTERPOLATION_PARAMETERS_-
LINE_GET
FIELD_ROUTINES, 347
FIELD_IO_BASIS_LHTP_FAMILY_LABEL
FIELD_IO_ROUTINES, 315
FIELD_IO_COMPONENT_LABEL
FIELD_IO_ROUTINES, 330
FIELD_IO_CREATE_FIELDS
FIELD_IO_ROUTINES, 315
FIELD_IO_DERIVATIVE_INFO
FIELD_IO_ROUTINES, 316
FIELD_IO_DERIVATIVE_LABEL
FIELD_IO_ROUTINES, 330
FIELD_IO_ELEMENTAL_INFO_SET_-
ATTACH_LOCAL_PROCESS
FIELD_IO_ROUTINES, 316
FIELD_IO_ELEMENTAL_INFO_SET_-
FINALIZE
FIELD_IO_ROUTINES, 316
FIELD_IO_ELEMENTAL_INFO_SET_-
INITIALISE
FIELD_IO_ROUTINES, 317
FIELD_IO_ELEMENTAL_INFO_SET_SORT
FIELD_IO_ROUTINES, 317
FIELD_IO_ELEMENTS_EXPORT
FIELD_IO_ROUTINES, 317
FIELD_IO_EXPORT_ELEMENTAL_GROUP_-
HEADER_FORTRAN
FIELD_IO_ROUTINES, 318
FIELD_IO_EXPORT_ELEMENTS_INTO_-
FIELD_IO_ROUTINES, 318
FIELD_IO_EXPORT_NODAL_GROUP_-
HEADER_FORTRA
FIELD_IO_ROUTINES, 319
FIELD_IO_EXPORT_NODES_INTO_LOC
FIELD_IO_ROUTINES, 319
FIELD_IO_FIELD_INFO
FIELD_IO_ROUTINES, 320
FIELD_IO_FIELD_LABEL
FIELD_IO_ROUTINES, 330
FIELD_IO_FILEDS_GROUP_INFO_GET
FIELD_IO_ROUTINES, 320

FIELD_IO_FILEDS_IMPORT
 FIELD_IO_ROUTINES, 321
FIELD_IO_FILL_BASIS_INFO
 FIELD_IO_ROUTINES, 321
FIELD_IO_FORTRAN_FILE_CLOSE
 FIELD_IO_ROUTINES, 322
FIELD_IO_FORTRAN_FILE_OPEN
 FIELD_IO_ROUTINES, 322
FIELD_IO_FORTRAN_FILE_READ_DP
 FIELD_IO_ROUTINES, 322
FIELD_IO_FORTRAN_FILE_READ_INTG
 FIELD_IO_ROUTINES, 323
FIELD_IO_FORTRAN_FILE_READ_STRING
 FIELD_IO_ROUTINES, 323
FIELD_IO_FORTRAN_FILE_REWIND
 FIELD_IO_ROUTINES, 324
FIELD_IO_FORTRAN_FILE_WRITE_DP
 FIELD_IO_ROUTINES, 324
FIELD_IO_FORTRAN_FILE_WRITE_INTG
 FIELD_IO_ROUTINES, 324
FIELD_IO_FORTRAN_FILE_WRITE_STRING
 FIELD_IO_ROUTINES, 325
FIELD_IO_IMPORT_GLOBAL_MESH
 FIELD_IO_ROUTINES, 325
FIELD_IO_LABEL_DERIVATIVE_INFO_GET
 FIELD_IO_ROUTINES, 326
FIELD_IO_LABEL_FIELD_INFO_GET
 FIELD_IO_ROUTINES, 327
FIELD_IO_MULTI_FILES_INFO_GET
 FIELD_IO_ROUTINES, 327
FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS
 FIELD_IO_ROUTINES, 327
FIELD_IO_NODAL_INFO_SET_FINALIZE
 FIELD_IO_ROUTINES, 328
FIELD_IO_NODAL_INFO_SET_INITIALISE
 FIELD_IO_ROUTINES, 328
FIELD_IO_NODAL_INFO_SET_SORT
 FIELD_IO_ROUTINES, 328
FIELD_IO_NODES_EXPORT
 FIELD_IO_ROUTINES, 329
FIELD_IO_ROUTINES, 311
 FIE, 315
 FIELD_IO_BASIS_LHTP_FAMILY_-
 LABEL, 315
 FIELD_IO_COMPONENT_LABEL, 330
 FIELD_IO_CREATE_FIELDS, 315
 FIELD_IO_DERIVATIVE_INFO, 316
 FIELD_IO_DERIVATIVE_LABEL, 330
 FIELD_IO_ELEMENTAL_INFO_SET_-
 ATTACH_LOCAL_PROCESS, 316
 FIELD_IO_ELEMENTAL_INFO_SET_-
 FINALIZE, 316
 FIELD_IO_ELEMENTAL_INFO_SET_-
 INITIALISE, 317
 FIELD_IO_ELEMENTAL_INFO_SET_-
 SORT, 317
FIELD_IO_ELEMENTS_EXPORT, 317
**FIELD_IO_EXPORT_ELEMENTAL_-
 GROUP_HEADER_FORTRAN**, 318
FIELD_IO_EXPORT_ELEMENTS_INTO_,
 318
**FIELD_IO_EXPORT_NODAL_GROUP_-
 HEADER_FORTRA**, 319
FIELD_IO_EXPORT_NODES_INTO_LOC,
 319
FIELD_IO_FIELD_INFO, 320
FIELD_IO_FIELD_LABEL, 330
FIELD_IO_FILEDS_GROUP_INFO_GET,
 320
FIELD_IO_FILEDS_IMPORT, 321
FIELD_IO_FILL_BASIS_INFO, 321
FIELD_IO_FORTRAN_FILE_CLOSE, 322
FIELD_IO_FORTRAN_FILE_OPEN, 322
FIELD_IO_FORTRAN_FILE_READ_DP,
 322
FIELD_IO_FORTRAN_FILE_READ_INTG,
 323
FIELD_IO_FORTRAN_FILE_READ_STRING,
 323
FIELD_IO_FORTRAN_FILE_REWIND,
 324
FIELD_IO_FORTRAN_FILE_WRITE_DP,
 324
FIELD_IO_IMPORT_GLOBAL_MESH,
 325
FIELD_IO_LABEL_DERIVATIVE_INFO_GET,
 326
FIELD_IO_LABEL_FIELD_INFO_GET,
 327
FIELD_IO_MULTI_FILES_INFO_GET,
 327
FIELD_IO_NODAL_INFO_SET_ATTACH_LOCAL_PROCESS,
 327
FIELD_IO_NODAL_INFO_SET_FINALIZE,
 328
FIELD_IO_NODAL_INFO_SET_INITIALISE,
 328
FIELD_IO_NODAL_INFO_SET_SORT, 328
FIELD_IO_NODES_EXPORT, 329
**FIELD_IO_SCALE_FACTORS_NUMBER_-
 TYPE**, 331
**FIELD_IO_SCALE_FACTORS_-
 PROPERTY_TYPE**, 331
**FIELD_IO_TRANSLATE_LABEL_INTO_-
 INTERPOLATION_TYPE**, 329
FIELD_IO_VARIABLE_LABEL, 331

SHAPE_SIZE, 331
STRING_TO_MUTI_INTEGERS_VS, 330
STRING_TO_MUTI_REALS_VS, 330
FIELD_IO_ROUTINES::FIELD_IO_-
ELEMENTALL_INFO_SET, 761
ELEMENTAL_INFO_SET, 761
FIELDS, 761
LIST_OF_GLOBAL_NUMBER, 761
NUMBER_OF_ELEMENTS, 762
FIELD_IO_ROUTINES::FIELD_IO_INFO_SET,
763
COMPONENTS, 763
NUMBER_OF_COMPONENTS, 763
SAME_HEADER, 763
FIELD_IO_ROUTINES::FIELD_IO_NODAL_-
INFO_SET, 764
FIELDS, 764
LIST_OF_GLOBAL_NUMBER, 764
NODAL_INFO_SET, 764
NUMBER_OF_NODES, 764
FIELD_IO_ROUTINES::FIELD_VARIABLE_-
COMPONENT_PTR_TYPE, 766
PTR, 766
FIELD_IO_ROUTINES::MESH_ELEMENTS_-
TYPE_PTR_TYPE, 767
PTR, 767
FIELD_IO_SCALE_FACTORS_NUMBER_TYPE
FIELD_IO_ROUTINES, 331
FIELD_IO_SCALE_FACTORS_PROPERTY_-
TYPE
FIELD_IO_ROUTINES, 331
FIELD_IO_TRANSLATE_LABEL_INTO_-
INTERPOLATION_TYPE
FIELD_IO_ROUTINES, 329
FIELD_IO_VARIABLE_LABEL
FIELD_IO_ROUTINES, 331
FIELD_MATERIAL_TYPE
FIELD_ROUTINES_FieldTypes, 61
FIELD_MESH_DECOMPOSITION_SET_-
NUMBER
FIELD_ROUTINES::FIELD_MESH_-
DECOMPOSITION_SET, 773
FIELD_MESH_DECOMPOSITION_SET_PTR
FIELD_ROUTINES::FIELD_MESH_-
DECOMPOSITION_SET, 773
FIELD_NO_SCALING
FIELD_ROUTINES_ScalingTypes, 72
FIELD_NODE_BASED_INTERPOLATION
FIELD_ROUTINES_InterpolationTypes, 63
FIELD_NODE_DOF_TYPE
FIELD_ROUTINES_DofTypes, 67
FIELD_NORMAL_VARIABLE_TYPE
FIELD_ROUTINES_VariableTypes, 64
FIELD_NUMBER_OF_COMPONENTS_SET_-
NUMBER
FIELD_ROUTINES::FIELD_NUMBER_-
OF_COMPONENTS_SET, 774
FIELD_NUMBER_OF_COMPONENTS_SET_-
PTR
FIELD_ROUTINES::FIELD_NUMBER_-
OF_COMPONENTS_SET, 774
FIELD_NUMBER_OF_SET_TYPES
FIELD_ROUTINES_ParameterSetTypes, 70
FIELD_NUMBER_OF_VARIABLE_TYPES
FIELD_ROUTINES_VariableTypes, 65
FIELD_NUMBER_OF_VARIABLES_SET_-
NUMBER
FIELD_ROUTINES::FIELD_NUMBER_-
OF_VARIABLES_SET, 775
FIELD_NUMBER_OF_VARIABLES_SET_PTR
FIELD_ROUTINES::FIELD_NUMBER_-
OF_VARIABLES_SET, 775
FIELD_POINT_BASED_INTERPOLATION
FIELD_ROUTINES_InterpolationTypes, 63
FIELD_POINT_DOF_TYPE
FIELD_ROUTINES_DofTypes, 68
FIELD_ROUTINES, 332
FI, 336
FIELD_COMPONENT_INTER, 337
FIELD_COMPONENT_MESH_COM, 337
FIELD_CREATE_FINISH, 338
FIELD_CREATE_VALUES_CACHE_-
FINALISE, 338
FIELD_CREATE_VALUES_CACHE_-
INITIALISE, 339
FIELD_DEPENDENT_TYPE_SET_-
NUMBER, 339
FIELD_DEPENDENT_TYPE_SET_PTR,
340
FIELD_DESTROY, 340
FIELD_DIMENSION_SET_NUMBER, 341
FIELD_DIMENSION_SET_PTR, 341
FIELD_INTERPOLATE_GAUSS, 342
FIELD_INTERPOLATE_XI, 342
FIELD_INTERPOLATED_POINT_-
FINALISE, 343
FIELD_INTERPOLATED_POINT_-
INITIALISE, 343
FIELD_INTERPOLATED_POINT_-
METRICS_CALCULATE, 344
FIELD_INTERPOLATED_POINT_-
METRICS_FINALISE, 344
FIELD_INTERPOLATED_POINT_-
METRICS_INITIALISE, 345
FIELD_INTERPOLATION_-
PARAMETERS_ELEMENT_GET,
345

**FIELD_INTERPOLATION_-
PARAMETERS_FINALISE**, 346
**FIELD_INTERPOLATION_-
PARAMETERS_INITIALISE**, 347
**FIELD_INTERPOLATION_-
PARAMETERS_LINE_GET**, 347
**FIELD_VARIABLE_COMPONENT_-
FINALISE**, 348
**FIELD_VARIABLE_COMPONENT_-
INITIALISE**, 348
FIELD_VARIABLES_FINALISE, 349
FIELD_VARIABLES_INITIALISE, 349
FIELDS_FINALISE, 350
FIELDS_INITIALISE, 350
FIELD_ROUTINES::DependentTypes, 58
FIELD_ROUTINES::DimensionTypes, 59
FIELD_ROUTINES::DofTypes, 67
**FIELD_ROUTINES::FIELD_COMPONENT_-
INTERPOLATION_SET**, 768
**FIELD_COMPONENT_INTERPOLATION_-
SET_NUMBER**, 768
**FIELD_COMPONENT_INTERPOLATION_-
SET_PTR**, 768
**FIELD_ROUTINES::FIELD_COMPONENT_-
MESH_COMPONENT_SET**, 769
**FIELD_COMPONENT_MESH_-
COMPONENT_SET_NUMBER**, 769
**FIELD_COMPONENT_MESH_-
COMPONENT_SET_PTR**, 769
**FIELD_ROUTINES::FIELD_DEPENDENT_-
TYPE_SET**, 770
**FIELD_DEPENDENT_TYPE_SET_-
NUMBER**, 770
FIELD_DEPENDENT_TYPE_SET_PTR,
770
FIELD_ROUTINES::FIELD_DIMENSION_SET,
771
FIELD_DIMENSION_SET_NUMBER, 771
FIELD_DIMENSION_SET_PTR, 771
**FIELD_ROUTINES::FIELD_GEOMETRIC_-
FIELD_SET**, 772
**FIELD_GEOMETRIC_FIELD_SET_-
NUMBER**, 772
FIELD_GEOMETRIC_FIELD_SET_PTR,
772
**FIELD_ROUTINES::FIELD_MESH_-
DECOMPOSITION_SET**, 773
**FIELD_MESH_DECOMPOSITION_SET_-
NUMBER**, 773
**FIELD_MESH_DECOMPOSITION_SET_-
PTR**, 773
**FIELD_ROUTINES::FIELD_NUMBER_OF_-
COMPONENTS_SET**, 774
**FIELD_NUMBER_OF_COMPONENTS_-
SET_NUMBER**, 774
**FIELD_NUMBER_OF_COMPONENTS_-
SET_PTR**, 774
**FIELD_ROUTINES::FIELD_NUMBER_OF_-
VARIABLES_SET**, 775
**FIELD_NUMBER_OF_VARIABLES_SET_-
NUMBER**, 775
**FIELD_NUMBER_OF_VARIABLES_SET_-
PTR**, 775
**FIELD_ROUTINES::FIELD_SCALING_TYPE_-
SET**, 776
FIELD_SCALING_TYPE_SET_NUMBER,
776
FIELD_SCALING_TYPE_SET_PTR, 776
FIELD_ROUTINES::FIELD_TYPE_SET, 777
FIELD_TYPE_SET_NUMBER, 777
FIELD_TYPE_SET_PTR, 777
FIELD_ROUTINES::FieldTypes, 60
FIELD_ROUTINES::InterpolationTypes, 62
FIELD_ROUTINES::ParameterSetTypes, 69
FIELD_ROUTINES::ScalingTypes, 71
FIELD_ROUTINES::VariableTypes, 64
FIELD_ROUTINES_DependentTypes
FIELD_DEPENDENT_TYPE, 58
FIELD_INDEPENDENT_TYPE, 58
FIELD_ROUTINES_DimensionTypes
FIELD_SCALAR_DIMENSION_TYPE, 59
FIELD_VECTOR_DIMENSION_TYPE, 59
FIELD_ROUTINES_DofTypes
FIELD_CONSTANT_DOF_TYPE, 67
FIELD_ELEMENT_DOF_TYPE, 67
FIELD_NODE_DOF_TYPE, 67
FIELD_POINT_DOF_TYPE, 68
FIELD_ROUTINES_FieldTypes
FIELD_FIBRE_TYPE, 60
FIELD_GENERAL_TYPE, 60
FIELD_GEOMETRIC_TYPE, 61
FIELD_MATERIAL_TYPE, 61
FIELD_ROUTINES_InterpolationTypes
FIELD_CONSTANT_INTERPOLATION, 62
**FIELD_ELEMENT_BASED_-
INTERPOLATION**, 62
FIELD_NODE_BASED_INTERPOLATION,
63
FIELD_POINT_BASED_INTERPOLATION,
63
FIELD_ROUTINES_ParameterSetTypes
**FIELD_BOUNDARY_CONDITIONS_SET_-
TYPE**, 69
**FIELD_INITIAL_CONDITIONS_SET_-
TYPE**, 69
FIELD_NUMBER_OF_SET_TYPES, 70
FIELD_VALUES_SET_TYPE, 70

FIELD_ROUTINES_ScalingTypes
FIELD_ARC_LENGTH_SCALING, 71
FIELD_ARITHMETIC_MEAN_SCALING,
71
FIELD_HARMONIC_MEAN_SCALING, 72
FIELD_NO_SCALING, 72
FIELD_UNIT_SCALING, 72
FIELD_ROUTINES_VariableTypes
FIELD_NORMAL_VARIABLE_TYPE, 64
FIELD_NUMBER_OF_VARIABLE_TYPES,
65
FIELD_STANDARD_VARIABLE_TYPE, 65
FIELD_TIME_DERIV1_VARIABLE_TYPE,
65
FIELD_TIME_DERIV2_VARIABLE_TYPE,
65
FIELD_SCALAR_DIMENSION_TYPE
FIELD_ROUTINES_DimensionTypes, 59
FIELD_SCALING_TYPE_SET_NUMBER
FIELD_ROUTINES::FIELD_SCALING_-
TYPE_SET, 776
FIELD_SCALING_TYPE_SET_PTR
FIELD_ROUTINES::FIELD_SCALING_-
TYPE_SET, 776
FIELD_STANDARD_VARIABLE_TYPE
FIELD_ROUTINES_VariableTypes, 65
FIELD_TIME_DERIV1_VARIABLE_TYPE
FIELD_ROUTINES_VariableTypes, 65
FIELD_TIME_DERIV2_VARIABLE_TYPE
FIELD_ROUTINES_VariableTypes, 65
FIELD_TYPE_SET_NUMBER
FIELD_ROUTINES::FIELD_TYPE_SET,
777
FIELD_TYPE_SET_PTR
FIELD_ROUTINES::FIELD_TYPE_SET,
777
FIELD_UNIT_SCALING
FIELD_ROUTINES_ScalingTypes, 72
FIELD_VALUES_SET_TYPE
FIELD_ROUTINES_ParameterSetTypes, 70
FIELD_VARIABLE
TYPES::FIELD_INTERPOLATION_-
PARAMETERS_TYPE, 1093
TYPES::FIELD_VARIABLE_-
COMPONENT_TYPE, 1113
FIELD_VARIABLE_COMPONENT_FINALISE
FIELD_ROUTINES, 348
FIELD_VARIABLE_COMPONENT_INITIALISE
FIELD_ROUTINES, 348
FIELD_VARIABLES_FINALISE
FIELD_ROUTINES, 349
FIELD_VARIABLES_INITIALISE
FIELD_ROUTINES, 349
FIELD_VECTOR_DIMENSION_TYPE
FIELD_ROUTINES_DimensionTypes, 59
FIELDS
FIELD_IO_ROUTINES::FIELD_IO_-
ELEMENTALL_INFO_SET, 761
FIELD_IO_ROUTINES::FIELD_IO_-
NODAL_INFO_SET, 764
TYPES::FIELD_TYPE, 1110
TYPES::FIELDS_TYPE, 1119
TYPES::REGION_TYPE, 1171
FIELDS_FINALISE
FIELD_ROUTINES, 350
FIELDS_INITIALISE
FIELD_ROUTINES, 350
FIELDS_USING
TYPES::FIELD_GEOMETRIC_-
PARAMETERS_TYPE, 1086
FILE_BEGINNING
BINARY_FILE, 189
FILE_CHANGE_ENDIAN
BINARY_FILE, 189
FILE_CURRENT
BINARY_FILE, 189
FILE_END
BINARY_FILE, 189
FILE_INFORMATION
BINARY_FILE::BINARY_FILE_TYPE, 701
FILE_NAME
BINARY_FILE::BINARY_FILE_INFO_-
TYPE, 699
FILE_NUMBER
BINARY_FILE::BINARY_FILE_INFO_-
TYPE, 699
FILE SAME_ENDIAN
BINARY_FILE, 189
FINITE_ELEMENT_ROUTINES, 352
FEM_ELEMENT_MATRICES_FINALISE,
352
FEM_ELEMENT_MATRICES_INITIALISE,
352
FIXED_CONDITIONS
TYPES::EQUATIONS_SET_TYPE, 1066
FIXED_CONDITIONS_FINISHED
TYPES::EQUATIONS_SET_FIXED_-
CONDITIONS_TYPE, 1056
FLAG_ERROR_C
BASE_ROUTINES, 167
BASE_ROUTINES::FLAG_ERROR, 687
FLAG_ERROR_VS
BASE_ROUTINES, 167
BASE_ROUTINES::FLAG_ERROR, 687
FLAG_WARNING_C
BASE_ROUTINES, 167
BASE_ROUTINES::FLAG_WARNING, 689
FLAG_WARNING_VS

BASE_ROUTINES, 168
 BASE_ROUTINES::FLAG_WARNING, 689
 FLIPENDIAN
 binary_file_c.c, 1219
 FLOATTYPE
 binary_file_c.c, 1219
 FLUID_MECHANICS_ROUTINES, 353
 FOCUS
 TYPES::COORDINATE_SYSTEM_TYPE, 959
 FROM_DIAG_TYPE
 BASE_ROUTINES_DiagnosticTypes, 26
 FROM_TIMING_TYPE
 BASE_ROUTINES_TimingTypes, 28
 FSTRINGLENGTH
 F90C, 310
 GAUSS_BASIS_FNS
 TYPES::QUADRATURE_SCHEME_TYPE, 1165
 GAUSS_ORDER
 TYPES::QUADRATURE_TYPE, 1167
 GAUSS_POSITIONS
 TYPES::QUADRATURE_SCHEME_TYPE, 1165
 GAUSS_WEIGHTS
 TYPES::QUADRATURE_SCHEME_TYPE, 1166
 GENERAL_OUTPUT_TYPE
 BASE_ROUTINES_OutputType, 20
 GENERATED_MESH
 TYPES::GENERATED_MESH_-
 REGULAR_TYPE, 1120
 GENERATED_MESH_CREATE_FINISH
 GENERATED_MESH_ROUTINES, 355
 GENERATED_MESH_CREATE_START
 GENERATED_MESH_ROUTINES, 355
 GENERATED_MESH_DESTROY
 GENERATED_MESH_ROUTINES, 355
 GENERATED_MESH_FINALISE
 GENERATED_MESH_ROUTINES, 356
 GENERATED_MESH_FRACTAL_TREE_-
 MESH_TYPE
 GENERATED_MESH_ROUTINES_-
 GeneratedMeshTypes, 73
 GENERATED_MESH_INITALISE
 GENERATED_MESH_ROUTINES, 356
 GENERATED_MESH_POLAR_MESH_TYPE
 GENERATED_MESH_ROUTINES_-
 GeneratedMeshTypes, 73
 GENERATED_MESH_REGULAR_MESH_TYPE
 GENERATED_MESH_ROUTINES_-
 GeneratedMeshTypes, 73
 GENERATED_MESH_ROUTINES, 354
 GENERATED_MESH_CREATE_FINISH, 355
 GENERATED_MESH_CREATE_START, 355
 GENERATED_MESH_DESTROY, 355
 GENERATED_MESH_FINALISE, 356
 GENERATED_MESH_INITALISE, 356
 GENERATED_MESH_TYPE_SET, 356
 GENERATED_MESH_-
 ROUTINES::GeneratedMeshTypes, 73
 GENERATED_MESH_ROUTINES_-
 GeneratedMeshTypes
 GENERATED_MESH_FRACTAL_TREE_-
 MESH_TYPE, 73
 GENERATED_MESH_POLAR_MESH_-
 TYPE, 73
 GENERATED_MESH_REGULAR_MESH_-
 TYPE, 73
 GENERATED_MESH_TYPE_SET
 GENERATED_MESH_ROUTINES, 356
 GENERATED_TYPE
 TYPES::GENERATED_MESH_TYPE, 1122
 GEOMETRIC_FIELD
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 TYPES::EQUATIONS_SET_GEOMETRY_-
 TYPE, 1058
 TYPES::FIELD_TYPE, 1110
 GEOMETRIC_FIELD_PARAMETERS
 TYPES::FIELD_TYPE, 1110
 GEOMETRIC_INTERP_PARAMETERS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 GEOMETRIC_INTERP_POINT
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1034
 GEOMETRIC_INTERP_POINT_METRICS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
 GEOMETRY
 TYPES::EQUATIONS_SET_TYPE, 1066
 get_
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::get, 823
 GET_BUFFER_LEN
 ISO_VARYING_STRING, 451
 get_set_CH
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::get, 823
 get_set_VS
 ISO_VARYING_STRING, 440
 ISO_VARYING_STRING::get, 823
 get_unit

ISO_VARYING_STRING, 440
ISO_VARYING_STRING::get, 823
get_unit_set_CH
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::get, 823
get_unit_set_VS
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::get, 823
GHOST_LIST
 TYPES::DOMAIN_MAPPING_TYPE, 1013
GHOSTING_TYPE
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 980
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 992
GL
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
GLOBAL_BOUNDARY_CONDITIONS
 TYPES::EQUATIONS_SET_FIXED_-
 CONDITIONS_TYPE, 1056
GLOBAL_COORDINATE_SYSTEM
 COORDINATE_ROUTINES, 260
GLOBAL_DOF_LIST
 TYPES::FIELD_VARIABLE_TYPE, 1117
GLOBAL_DOF_OFFSET
 TYPES::FIELD_VARIABLE_TYPE, 1117
GLOBAL_ELEMENT_NODES
 TYPES::MESH_ELEMENT_TYPE, 1135
GLOBAL_M
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977
GLOBAL_N
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977
 TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, 985
GLOBAL_NUMBER
 TYPES::BASIS_TYPE, 953
 TYPES::DECOMPOSITION_ELEMENT_-
 TYPE, 962
 TYPES::DECOMPOSITION_TYPE, 971
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::EQUATIONS_SET_TYPE, 1066
 TYPES::FIELD_TYPE, 1110
 TYPES::MESH_ELEMENT_TYPE, 1135
 TYPES::MESH_NODE_TYPE, 1138
 TYPES::MESH_TYPE, 1146
 TYPES::NODE_TYPE, 1150
 TYPES::PROBLEM_TYPE, 1161
 TYPES::QUADRATURE_SCHEME_TYPE,
 1166
GLOBAL_NUMBERS

TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, 986
GLOBAL_REGION
 REGION_ROUTINES, 608
GLOBAL_ROW_NUMBERS
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977
GLOBAL_TO_LOCAL_MAP
 TYPES::DOMAIN_MAPPING_TYPE, 1013
GU
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
HEAD
 BASE_ROUTINES::ROUTINE_LIST_TYPE,
 693
HEADER
 BINARY_FILE::BINARY_TAG_TYPE, 702
HEAP_SORT_DP
 SORTING, 643
 SORTING::HEAP_SORT, 921
HEAP_SORT_INTG
 SORTING, 643
 SORTING::HEAP_SORT, 921
HEAP_SORT_SP
 SORTING, 643
 SORTING::HEAP_SORT, 921
Hello, 1
HELP_OUTPUT_TYPE
 BASE_ROUTINES_OutputType, 20
HERMITE
 TYPES::BASIS_TYPE, 953
I0_DP
 MATHS, 488
 MATHS::I0, 876
I0_SP
 MATHS, 488
 MATHS::I0, 876
I1_DP
 MATHS, 488
 MATHS::I1, 877
I1_SP
 MATHS, 488
 MATHS::I1, 877
iachar_
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::iachar, 825
ichar_
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::ichar, 826
ID
 TYPES::MATRIX_TYPE, 1131
 TYPES::VECTOR_TYPE, 1207

IN_DIAG_TYPE
 BASE_ROUTINES_DiagnosticTypes, 26

IN_TIMING_TYPE
 BASE_ROUTINES_TimingTypes, 28

INCLUSIVE_CPU_TIME
 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695

INCLUSIVE_SYSTEM_TIME
 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695

INDEX
 BINARY_FILE::BINARY_TAG_TYPE, 702

index_CH_VS
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::index, 827

index_VS_CH
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::index, 827

index_VS_VS
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::index, 827

INITIAL_POSITION
 TYPES::NODE_TYPE, 1150

INITIAL_SIZE
 LISTS::LIST_TYPE, 868

INPUT_OUTPUT, 358
 WR, 372–392
 WRITE_STRING_C, 393
 WRITE_STRING_FMT, 393–397
 WRITE_STRING_FMT_VALUE_C, 398
 WRITE_STRING_FMT_VALUE_DP, 398
 WRITE_STRING_FMT_VALUE_INTG, 399
 WRITE_STRING_FMT_VALUE_L, 399
 WRITE_STRING_FMT_VALUE_LINTG,
 400
 WRITE_STRING_FMT_VALUE_SP, 400
 WRITE_STRING_FMT_VALUE_VS, 401
 WRITE_STRING_IDX, 401–405
 WRITE_STRING_MATRIX_DP, 405
 WRITE_STRING_MATRIX_INTG, 407
 WRITE_STRING_MATRIX_L, 408
 WRITE_STRING_MATRIX_LINTG, 409
 WRITE_STRING_MATRIX_SP, 410
 WRITE_STRING_TWO_VALUE_C_C, 411
 WRITE_STRING_TWO_VALUE_C_DP, 412
 WRITE_STRING_TWO_VALUE_C_INTG,
 413
 WRITE_STRING_TWO_VALUE_C_L, 413
 WRITE_STRING_TWO_VALUE_C_SP, 414
 WRITE_STRING_TWO_VALUE_C_VS, 414
 WRITE_STRING_TWO_VALUE_DP_C, 415
 WRITE_STRING_TWO_VALUE_DP_DP,
 416

WRITE_STRING_TWO_VALUE_DP_INTG,
 416

WRITE_STRING_TWO_VALUE_DP_L, 417

WRITE_STRING_TWO_VALUE_DP_SP,
 417

WRITE_STRING_TWO_VALUE_DP_VS,
 418

WRITE_STRING_TWO_VALUE_INTG_C,
 418

WRITE_STRING_TWO_VALUE_INTG_DP,
 419

WRITE_STRING_TWO_VALUE_INTG_-
 INTG, 420

WRITE_STRING_TWO_VALUE_INTG_L,
 420

WRITE_STRING_TWO_VALUE_INTG_SP,
 421

WRITE_STRING_TWO_VALUE_INTG_VS,
 421

WRITE_STRING_TWO_VALUE_L_C, 422

WRITE_STRING_TWO_VALUE_L_DP, 423

WRITE_STRING_TWO_VALUE_L_INTG,
 423

WRITE_STRING_TWO_VALUE_L_L, 424

WRITE_STRING_TWO_VALUE_L_SP, 424

WRITE_STRING_TWO_VALUE_L_VS, 425

WRITE_STRING_TWO_VALUE_SP_C, 425

WRITE_STRING_TWO_VALUE_SP_DP,
 426

WRITE_STRING_TWO_VALUE_SP_INTG,
 427

WRITE_STRING_TWO_VALUE_SP_L, 427

WRITE_STRING_TWO_VALUE_SP_SP,
 428

WRITE_STRING_TWO_VALUE_SP_VS,
 428

WRITE_STRING_TWO_VALUE_VS_C, 429

WRITE_STRING_TWO_VALUE_VS_DP,
 430

WRITE_STRING_TWO_VALUE_VS_INTG,
 430

WRITE_STRING_TWO_VALUE_VS_L, 431

WRITE_STRING_TWO_VALUE_VS_SP,
 431

WRITE_STRING_TWO_VALUE_VS_VS,
 432

WRITE_STRING_VALUE_C, 433

WRITE_STRING_VALUE_DP, 433

WRITE_STRING_VALUE_INTG, 434

WRITE_STRING_VALUE_L, 434

WRITE_STRING_VALUE_LINTG, 435

WRITE_STRING_VALUE_SP, 435

WRITE_STRING_VALUE_VS, 436

WRITE_STRING_VS, 436

`INPUT_OUTPUT::MatrixNameIndexFormat`, [75](#)
`INPUT_OUTPUT::WRITE_STRING`, [778](#)
 `WRITE_STRING_C`, [778](#)
 `WRITE_STRING_VS`, [778](#)
`INPUT_OUTPUT::WRITE_STRING_FMT_-_TWO_VALUE`, [779](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_C`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_DP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_INTG`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_L`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_SP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_C_-_VS`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_C`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_DP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_INTG`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_L`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_SP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_DP_VS`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_C`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_DP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_INTG`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_L`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_SP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_D_-_INTG_VS`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_C`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_DP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_INTG`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_L`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_SP`, [781](#)
 `WRITE_STRING_FMT_TWO_VALUE_L_-_VS`, [781](#)
 `WRITE_STRING_TWO_VALUE_C_C`, [793](#)
 `WRITE_STRING_TWO_VALUE_C_DP`, [794](#)

WRITE_STRING_TWO_VALUE_C_INTG,
 794
 WRITE_STRING_TWO_VALUE_C_L, 794
 WRITE_STRING_TWO_VALUE_C_SP, 795
 WRITE_STRING_TWO_VALUE_C_VS, 795
 WRITE_STRING_TWO_VALUE_DP_C, 796
 WRITE_STRING_TWO_VALUE_DP_DP,
 796
 WRITE_STRING_TWO_VALUE_DP_INTG,
 797
 WRITE_STRING_TWO_VALUE_DP_L, 797
 WRITE_STRING_TWO_VALUE_DP_SP,
 798
 WRITE_STRING_TWO_VALUE_DP_VS,
 798
 WRITE_STRING_TWO_VALUE_INTG_C,
 799
 WRITE_STRING_TWO_VALUE_INTG_DP,
 799
 WRITE_STRING_TWO_VALUE_INTG_-
 INTG, 800
 WRITE_STRING_TWO_VALUE_INTG_L,
 800
 WRITE_STRING_TWO_VALUE_INTG_SP,
 801
 WRITE_STRING_TWO_VALUE_INTG_VS,
 801
 WRITE_STRING_TWO_VALUE_L_C, 802
 WRITE_STRING_TWO_VALUE_L_DP, 802
 WRITE_STRING_TWO_VALUE_L_INTG,
 803
 WRITE_STRING_TWO_VALUE_L_L, 803
 WRITE_STRING_TWO_VALUE_L_SP, 804
 WRITE_STRING_TWO_VALUE_L_VS, 804
 WRITE_STRING_TWO_VALUE_SP_C, 805
 WRITE_STRING_TWO_VALUE_SP_DP,
 805
 WRITE_STRING_TWO_VALUE_SP_INTG,
 806
 WRITE_STRING_TWO_VALUE_SP_L, 806
 WRITE_STRING_TWO_VALUE_SP_SP,
 807
 WRITE_STRING_TWO_VALUE_SP_VS,
 807
 WRITE_STRING_TWO_VALUE_VS_C, 808
 WRITE_STRING_TWO_VALUE_VS_DP,
 808
 WRITE_STRING_TWO_VALUE_VS_INTG,
 809
 WRITE_STRING_TWO_VALUE_VS_L, 809
 WRITE_STRING_TWO_VALUE_VS_SP,
 810
 WRITE_STRING_TWO_VALUE_VS_VS,
 810

INPUT_OUTPUT::WRITE_STRING_VALUE,
 812
 WRITE_STRING_VALUE_C, 812
 WRITE_STRING_VALUE_DP, 812
 WRITE_STRING_VALUE_INTG, 813
 WRITE_STRING_VALUE_L, 813
 WRITE_STRING_VALUE_LINTG, 813
 WRITE_STRING_VALUE_SP, 814
 WRITE_STRING_VALUE_VS, 814

INPUT_OUTPUT::WRITE_STRING_VECTOR,
 815
 WRITE_STRING_VECTOR_DP, 815
 WRITE_STRING_VECTOR_INTG, 815
 WRITE_STRING_VECTOR_L, 815
 WRITE_STRING_VECTOR_LINTG, 815
 WRITE_STRING_VECTOR_SP, 815

INPUT_OUTPUT_MatrixNameIndexFormat
 WRITE_STRING_MATRIX_NAME_AND_-
 INDICES, 75
 WRITE_STRING_MATRIX_NAME_ONLY,
 75

INQUIRE_EOF_BINARY_FILE
 BINARY_FILE, 178

INQUIRE_OPEN_BINARY_FILE
 BINARY_FILE, 178

insert_CH_CH
 ISO_VARYING_STRING, 441
 ISO_VARYING_STRING::insert, 828

insert_CH_VS
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::insert, 828

INSERT_TYPE
 TREES::TREE_TYPE, 944

insert_VS_CH
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::insert, 828

insert_VS_VS
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::insert, 828

INT_FORMAT
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699

integer
 f90c_c.c, 1272

INTEGER_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
 MACHINE_CONSTANTS, 481

INTEGERTYPE
 binary_file_c.c, 1219

INTERNAL_LIST
 TYPES::DOMAIN_MAPPING_TYPE, 1013

INTERPOLATED_POINT

TYPES::FIELD_INTERPOLATED_POINT_-
METRICS_TYPE, 1089

INTERPOLATION
TYPES::EQUATIONS_TYPE, 1078

INTERPOLATION_ORDER
TYPES::BASIS_TYPE, 953

INTERPOLATION_PARAMETERS
TYPES::FIELD_INTERPOLATED_POINT_-
TYPE, 1091

INTERPOLATION_TYPE
TYPES::BASIS_TYPE, 953
TYPES::FIELD_CREATE_VALUES_-
CACHE_TYPE, 1080
TYPES::FIELD_VARIABLE_-
COMPONENT_TYPE, 1113

INTERPOLATION_XI
TYPES::BASIS_TYPE, 953

INTG
KINDS_IntegerKinds, 77

INVERT_FULL_DP
MATHS, 488
MATHS::INVERT, 878

INVERT_FULL_SP
MATHS, 489
MATHS::INVERT, 878

IO1_FILE_UNIT
BASE_ROUTINES_FileUnits, 23

IO2_FILE_UNIT
BASE_ROUTINES_FileUnits, 23

IO3_FILE_UNIT
BASE_ROUTINES_FileUnits, 23

IO4_FILE_UNIT
BASE_ROUTINES_FileUnits, 24

IO5_FILE_UNIT
BASE_ROUTINES_FileUnits, 24

IS_ABBREVIATION_C_C
STRINGS, 650
STRINGS::IS_ABBREVIATION, 925

IS_ABBREVIATION_C_VS
STRINGS, 650
STRINGS::IS_ABBREVIATION, 925

IS_ABBREVIATION_VS_C
STRINGS, 651
STRINGS::IS_ABBREVIATION, 925

IS_ABBREVIATION_VS_VS
STRINGS, 651
STRINGS::IS_ABBREVIATION, 926

IS_DIGIT
STRINGS, 651

IS_LETTER
STRINGS, 651

IS_LOWERCASE
STRINGS, 651

IS_UPPERCASE

STRINGS, 652

IS_WHITESPACE
STRINGS, 652

ISBINARYFILEOPEN
BINARY_FILE::interface, 703

IsBinaryFileOpen
binary_file_c.c, 1221

ISDestroy
CMISS_PETSC::interface, 718

ISENDBINARYFILE
BINARY_FILE::interface, 704

IsEndBinaryFile
binary_file_c.c, 1221

ISLocalToGlobalMappingApply
CMISS_PETSC::interface, 718

ISLocalToGlobalMappingApplyIS
CMISS_PETSC::interface, 719

ISLocalToGlobalMappingCreate
CMISS_PETSC::interface, 719

ISLocalToGlobalMappingDestroy
CMISS_PETSC::interface, 719

ISLTGMAPPING
TYPES::DISTRIBUTED_MATRIX_-
PETSC_TYPE, 977
TYPES::DISTRIBUTED_VECTOR_-
PETSC_TYPE, 986

ISO_VARYING_STRING, 437
adjustl_, 439
adjustr_, 439
char_auto, 440
char_fixed, 440
extract_CH, 440
extract_VS, 440
get_, 440
GET_BUFFER_LEN, 451
get_set_CH, 440
get_set_VS, 440
get_unit, 440
get_unit_set_CH, 441
get_unit_set_VS, 441
iachar_, 441
ichar_, 441
index_CH_VS, 441
index_VS_CH, 441
index_VS_VS, 441
insert_CH_CH, 441
insert_CH_VS, 442
insert_VS_CH, 442
insert_VS_VS, 442
len_, 442
len_trim_, 442
lge_CH_VS, 442
lge_VS_CH, 442
lge_VS_VS, 442

lgt_CH_VS, 442
 lgt_VS_CH, 443
 lgt_VS_VS, 443
 lle_CH_VS, 443
 lle_VS_CH, 443
 lle_VS_VS, 443
 llt_CH_VS, 443
 llt_VS_CH, 443
 llt_VS_VS, 443
 op_assign_CH_VS, 443
 op_assign_VS_CH, 444
 op_concat_CH_VS, 444
 op_concat_VS_CH, 444
 op_concat_VS_VS, 444
 op_eq_CH_VS, 444
 op_eq_VS_CH, 444
 op_eq_VS_VS, 444
 op_ge_CH_VS, 445
 op_ge_VS_CH, 445
 op_ge_VS_VS, 445
 op_gt_CH_VS, 445
 op_gt_VS_CH, 445
 op_gt_VS_VS, 445
 op_le_CH_VS, 445
 op_le_VS_CH, 445
 op_le_VS_VS, 445
 op_lt_CH_VS, 446
 op_lt_VS_CH, 446
 op_lt_VS_VS, 446
 op_ne_CH_VS, 446
 op_ne_VS_CH, 446
 op_ne_VS_VS, 446
 put_CH, 446
 put_line_CH, 446
 put_line_unit_CH, 446
 put_line_unit_VS, 447
 put_line_VS, 447
 put_unit_CH, 447
 put_unit_VS, 447
 put_VS, 447
 remove_CH, 447
 remove_VS, 447
 repeat_, 447
 replace_CH_CH_auto, 447
 replace_CH_CH_CH_target, 448
 replace_CH_CH_fixed, 448
 replace_CH_CH_VS_target, 448
 replace_CH_VS_auto, 448
 replace_CH_VS_CH_target, 448
 replace_CH_VS_fixed, 448
 replace_CH_VS_VS_target, 448
 replace_VS_CH_auto, 449
 replace_VS_CH_CH_target, 449
 replace_VS_CH_fixed, 449
 replace_VS_CH_VS_target, 449
 replace_VS_VS_auto, 449
 replace_VS_VS_CH_target, 449
 replace_VS_VS_fixed, 449
 replace_VS_VS_VS_target, 450
 scan_CH_VS, 450
 scan_VS_CH, 450
 scan_VS_VS, 450
 split_CH, 450
 split_VS, 450
 trim_, 450
 var_str_, 451
 verify_CH_VS, 451
 verify_VS_CH, 451
 verify_VS_VS, 451
ISO_VARYING_STRING::adjustr, 816
 adjustr_, 816
ISO_VARYING_STRING::assignment(=), 817
 adjustl_, 817
 op_assign_CH_VS, 817
 op_assign_VS_CH, 817
 op_concat_CH_VS, 817
 op_concat_VS_CH, 818
 op_concat_VS_VS, 818
 op_eq_CH_VS, 818
 op_eq_VS_CH, 818
 op_eq_VS_VS, 818
 op_ge_CH_VS, 818
 op_ge_VS_CH, 818
 op_ge_VS_VS, 818
 op_gt_CH_VS, 818
 op_gt_VS_CH, 819
 op_gt_VS_VS, 819
 op_le_CH_VS, 819
 op_le_VS_CH, 819
 op_le_VS_VS, 819
 op_lt_CH_VS, 819
 op_lt_VS_CH, 819
 op_lt_VS_VS, 819
 op_ne_CH_VS, 819
 op_ne_VS_CH, 820
 op_ne_VS_VS, 820
ISO_VARYING_STRING::char, 821
 char_auto, 821
 char_fixed, 821
ISO_VARYING_STRING::extract, 822
 extract_CH, 822
 extract_VS, 822
ISO_VARYING_STRING::get, 823
 get_, 823
 get_set_CH, 823
 get_set_VS, 823
 get_unit, 823
 get_unit_set_CH, 823

get_unit_set_VS, 823
 ISO_VARYING_STRING::iachar, 825
 iachar_, 825
 ISO_VARYING_STRING::ichar, 826
 ichar_, 826
 ISO_VARYING_STRING::index, 827
 index_CH_VS, 827
 index_VS_CH, 827
 index_VS_VS, 827
 ISO_VARYING_STRING::insert, 828
 insert_CH_CH, 828
 insert_CH_VS, 828
 insert_VS_CH, 828
 insert_VS_VS, 828
 ISO_VARYING_STRING::len, 829
 len_, 829
 ISO_VARYING_STRING::len_trim, 830
 len_trim_, 830
 ISO_VARYING_STRING::lge, 831
 lge_CH_VS, 831
 lge_VS_CH, 831
 lge_VS_VS, 831
 ISO_VARYING_STRING::lgt, 832
 lgt_CH_VS, 832
 lgt_VS_CH, 832
 lgt_VS_VS, 832
 ISO_VARYING_STRING::lle, 833
 lle_CH_VS, 833
 lle_VS_CH, 833
 lle_VS_VS, 833
 ISO_VARYING_STRING::llt, 834
 llt_CH_VS, 834
 llt_VS_CH, 834
 llt_VS_VS, 834
 ISO_VARYING_STRING::put, 835
 put_CH, 835
 put_unit_CH, 835
 put_unit_VS, 835
 put_VS, 835
 ISO_VARYING_STRING::put_line, 836
 put_line_CH, 836
 put_line_unit_CH, 836
 put_line_unit_VS, 836
 put_line_VS, 836
 ISO_VARYING_STRING::remove, 837
 remove_CH, 837
 remove_VS, 837
 ISO_VARYING_STRING::repeat, 838
 repeat_, 838
 ISO_VARYING_STRING::replace, 839
 replace_CH_CH_auto, 839
 replace_CH_CH_CH_target, 839
 replace_CH_CH_fixed, 839
 replace_CH_CH_VS_target, 839
 replace_CH_VS_auto, 840
 replace_CH_VS_CH_target, 840
 replace_CH_VS_fixed, 840
 replace_CH_VS_VS_target, 840
 replace_VS_CH_auto, 840
 replace_VS_CH_CH_target, 840
 replace_VS_CH_fixed, 840
 replace_VS_CH_VS_target, 841
 replace_VS_VS_auto, 841
 replace_VS_VS_CH_target, 841
 replace_VS_VS_fixed, 841
 replace_VS_VS_VS_target, 841
 ISO_VARYING_STRING::scan, 842
 scan_CH_VS, 842
 scan_VS_CH, 842
 scan_VS_VS, 842
 ISO_VARYING_STRING::split, 843
 split_CH, 843
 split_VS, 843
 ISO_VARYING_STRING::trim, 844
 trim_, 844
 ISO_VARYING_STRING::var_str, 845
 var_str_, 845
 ISO_VARYING_STRING::VARYING_STRING,
 846
 chars, 846
 ISO_VARYING_STRING::verify, 847
 verify_CH_VS, 847
 verify_VS_CH, 847
 verify_VS_VS, 847
 ITERATIVE_PRECONDITIONER_TYPE
 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1125
 ITERATIVE_SOLVER
 TYPES::LINEAR_SOLVER_TYPE, 1127
 ITERATIVE_SOLVER_TYPE
 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1125
 JACOBIAN
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
 JACOBIAN_TYPE
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
 K0_DP
 MATHS, 489
 MATHS::K0, 879
 K0_SP
 MATHS, 489
 MATHS::K0, 879
 K1_DP
 MATHS, 489

MATHS::K1, 880
 K1_SP
 MATHS, 489
 MATHS::K1, 880
 KDP_DP
 MATHS, 489
 KDP_SP
 MATHS, 489
 KEY
 TREES::TREE_NODE_TYPE, 942
 KINDS, 452
 KINDS::ComplexKinds, 82
 KINDS::IntegerKinds, 77
 KINDS::RealKinds, 80
 KINDS_ComplexKinds
 _DP, 82
 _SP, 83
 DPC, 83
 SPC, 83
 KINDS_IntegerKinds
 INTG, 77
 LINTG, 77
 PTR, 77
 SINTG, 79
 KINDS_RealKinds
 DP, 80
 QP, 80
 SP, 80
 KSP
 TYPES::LINEAR_ITERATIVE_SOLVER_TYPE, 1125
 KSPCreate
 CMISS_PETSC::interface, 719
 KSPDestroy
 CMISS_PETSC::interface, 719
 KSPGetConvergedReason
 CMISS_PETSC::interface, 719
 KSPGetIterationNumber
 CMISS_PETSC::interface, 719
 KSPGetPC
 CMISS_PETSC::interface, 719
 KSPGetResidualNorm
 CMISS_PETSC::interface, 719
 KSPSetFromOptions
 CMISS_PETSC::interface, 719
 KSPSetOperators
 CMISS_PETSC::interface, 720
 KSPSetTolerances
 CMISS_PETSC::interface, 720
 KSPSetType
 CMISS_PETSC::interface, 720
 KSPSetUp
 CMISS_PETSC::interface, 720
 KSPSolve
 TREES::TREE_NODE_TYPE, 942
 CMISS_PETSC::interface, 720
 L2NORM_DP
 MATHS, 489
 MATHS::L2NORM, 881
 L2NORM_SP
 MATHS, 490
 MATHS::L2NORM, 881
 LABEL
 TYPES::NODE_TYPE, 1150
 TYPES::REGION_TYPE, 1171
 LAPACK, 453
 LAPACK::interface, 848
 DGESV, 848
 LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE
 LAPLACE_EQUATIONS_ROUTINES, 454
 LAPLACE_EQUATION_EQUATIONS_SETSETUP
 LAPLACE_EQUATIONS_ROUTINES, 455
 LAPLACE_EQUATION_EQUATIONS_SETSTANDARD_SETUP
 LAPLACE_EQUATIONS_ROUTINES, 455
 LAPLACE_EQUATION_EQUATIONS_SETSUBTYPE_SET
 LAPLACE_EQUATIONS_ROUTINES, 456
 LAPLACE_EQUATION_PROBLEM_SETUP
 LAPLACE_EQUATIONS_ROUTINES, 457
 LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP
 LAPLACE_EQUATIONS_ROUTINES, 457
 LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP
 LAPLACE_EQUATIONS_ROUTINES, 458
 LAPLACE_EQUATION_PROBLEM_SUBTYPE_SET
 LAPLACE_EQUATIONS_ROUTINES, 458
 LAPLACE_EQUATIONS_ROUTINES, 454
 LAPLACE_EQUATION_EQUATIONS_SETFINITE_ELEMENT_CALCULATE
 LAPLACE_EQUATIONS_ROUTINES, 454
 LAPLACE_EQUATION_EQUATIONS_SETSETUP, 455
 LAPLACE_EQUATION_EQUATIONS_SETSTANDARD_SETUP, 455
 LAPLACE_EQUATION_EQUATIONS_SETSUBTYPE_SET, 456
 LAPLACE_EQUATION_PROBLEM_SETUP, 457
 LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP, 457
 LAPLACE_EQUATION_PROBLEM_STANDARD_SETUP
 LAPLACE_EQUATIONS_ROUTINES, 458
 LEARN_FILE_UNIT
 BASE_ROUTINES_FileUnits, 24
 LEFT
 TREES::TREE_NODE_TYPE, 942

len_
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::len, 829

len_trim_
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::len_trim, 830

LENGTHS
 TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087

lge_CH_vs
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::lge, 831

lge_vs_ch
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::lge, 831

lge_vs_vs
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::lge, 831

lgt_CH_vs
 ISO_VARYING_STRING, 442
 ISO_VARYING_STRING::lgt, 832

lgt_vs_ch
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::lgt, 832

lgt_vs_vs
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::lgt, 832

LIBRARY_TYPE
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 980
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 992
 TYPES::SOLVER_MATRICES_TYPE, 1188

LINE_BASES
 TYPES::BASIS_TYPE, 953

LINEAR_DATA
 TYPES::EQUATIONS_TYPE, 1078

LINEAR_SOLVER
 TYPES::LINEAR_DIRECT_SOLVER_-
 TYPE, 1123
 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1125
 TYPES::SOLVER_TYPE, 1197

LINEAR_SOLVER_TYPE
 TYPES::LINEAR_SOLVER_TYPE, 1127

LINEARITY
 TYPES::EQUATIONS_SET_TYPE, 1066

LINES
 TYPES::DECOMPOSITION_LINES_TYPE,
 967
 TYPES::DECOMPOSITION_TOPOLOGY_-
 TYPE, 969
 TYPES::DOMAIN_LINES_TYPE, 1010

 TYPES::DOMAIN_TOPOLOGY_TYPE,
 1024

 LINTEGER_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699

 LINTG
 KINDS_IntegerKinds, 77

 LIST_BUBBLE_SORT_METHOD
 LISTSSortingMethod, 88

 LIST_CREATE_FINISH
 LISTS, 464

 LIST_CREATE_START
 LISTS, 464

 LIST_DATA_TYPE_SET
 LISTS, 465

 LIST_DESTROY
 LISTS, 465

 LIST_DETACH_AND_DESTROY_DP
 LISTS, 466
 LISTS::LIST_DETACH_AND_DESTROY,
 849

 LIST_DETACH_AND_DESTROY_INTG
 LISTS, 466
 LISTS::LIST_DETACH_AND_DESTROY,
 849

 LIST_DETACH_AND_DESTROY_SP
 LISTS, 467
 LISTS::LIST_DETACH_AND_DESTROY,
 850

 LIST_DP
 LISTS::LIST_TYPE, 868

 LIST_DP_TYPE
 LISTS_DataType, 84

 LIST_FINALISE
 LISTS, 467

 LIST_FINISHED
 LISTS::LIST_TYPE, 868

 LIST_HEAP_SORT_METHOD
 LISTSSortingMethod, 88

 LIST_INITIAL_SIZE_SET
 LISTS, 468

 LIST_INITIALISE
 LISTS, 468

 LIST_INTG
 LISTS::LIST_TYPE, 868

 LIST_INTG_TYPE
 LISTS_DataType, 84

 LIST_ITEM_ADD_DP1
 LISTS, 468
 LISTS::LIST_ITEM_ADD, 851

 LIST_ITEM_ADD_INTG1
 LISTS, 469
 LISTS::LIST_ITEM_ADD, 851

 LIST_ITEM_ADD_SP1

LISTS, 469
 LISTS::LIST_ITEM_ADD, 851
 LIST_ITEM_DELETE
 LISTS, 470
 LIST_ITEM_IN_LIST_DP1
 LISTS, 470
 LISTS::LIST_ITEM_IN_LIST, 853
 LIST_ITEM_IN_LIST_INTG1
 LISTS, 470
 LISTS::LIST_ITEM_IN_LIST, 853
 LIST_ITEM_IN_LIST_SP1
 LISTS, 471
 LISTS::LIST_ITEM_IN_LIST, 854
 LIST_NUMBER_OF_ITEMS_GET
 LISTS, 471
 LIST_OF_GLOBAL_NUMBER
 FIELD_IO_ROUTINES::FIELD_IO_-
 ELEMENTALL_INFO_SET, 761
 FIELD_IO_ROUTINES::FIELD_IO_-
 NODAL_INFO_SET, 764
 LIST_REMOVE_DUPLICATES
 LISTS, 472
 LIST_SEARCH_DP_ARRAY
 LISTS, 472
 LISTS::LIST_SEARCH, 856
 LIST_SEARCH_INTG_ARRAY
 LISTS, 473
 LISTS::LIST_SEARCH, 856
 LIST_SEARCH_LINEAR_DP_ARRAY
 LISTS, 473
 LISTS::LIST_SEARCH_LINEAR, 858
 LIST_SEARCH_LINEAR_INTG_ARRAY
 LISTS, 474
 LISTS::LIST_SEARCH_LINEAR, 858
 LIST_SEARCH_LINEAR_SP_ARRAY
 LISTS, 474
 LISTS::LIST_SEARCH_LINEAR, 858
 LIST_SEARCH_SP_ARRAY
 LISTS, 475
 LISTS::LIST_SEARCH, 857
 LIST_SHELL_SORT_METHOD
 LISTSSortingMethod, 88
 LIST_SORT_ASCENDING_TYPE
 LISTS_SortingOrder, 86
 LIST_SORT_BUBBLE_DP_ARRAY
 LISTS, 475
 LISTS::LIST_SORT_BUBBLE, 862
 LIST_SORT_BUBBLE_INTG_ARRAY
 LISTS, 475
 LISTS::LIST_SORT_BUBBLE, 862
 LIST_SORT_BUBBLE_SP_ARRAY
 LISTS, 476
 LISTS::LIST_SORT_BUBBLE, 862
 LIST_SORT_DESCENDING_TYPE

LISTS_SortingOrder, 86
 LIST_SORT_DP_ARRAY
 LISTS, 476
 LISTS::LIST_SORT, 860
 LIST_SORT_HEAP_DP_ARRAY
 LISTS, 476
 LISTS::LIST_SORT_HEAP, 864
 LIST_SORT_HEAP_INTG_ARRAY
 LISTS, 477
 LISTS::LIST_SORT_HEAP, 864
 LIST_SORT_HEAP_SP_ARRAY
 LISTS, 477
 LISTS::LIST_SORT_HEAP, 864
 LIST_SORT_INTG_ARRAY
 LISTS, 477
 LISTS::LIST_SORT, 860
 LIST_SORT_SHELL_DP_ARRAY
 LISTS, 478
 LISTS::LIST_SORT_SHELL, 866
 LIST_SORT_SHELL_INTG_ARRAY
 LISTS, 478
 LISTS::LIST_SORT_SHELL, 866
 LIST_SORT_SHELL_SP_ARRAY
 LISTS, 478
 LISTS::LIST_SORT_SHELL, 866
 LIST_SORT_SP_ARRAY
 LISTS, 479
 LISTS::LIST_SORT, 860
 LIST_SP
 LISTS::LIST_TYPE, 868
 LIST_SP_TYPE
 LISTS_DataType, 85
 LIST_TO_CHARACTER_C
 STRINGS, 652
 STRINGS::LIST_TO_CHARACTER, 927
 LIST_TO_CHARACTER_DP
 STRINGS, 653
 STRINGS::LIST_TO_CHARACTER, 927
 LIST_TO_CHARACTER_INTG
 STRINGS, 653
 STRINGS::LIST_TO_CHARACTER, 928
 LIST_TO_CHARACTER_L
 STRINGS, 654
 STRINGS::LIST_TO_CHARACTER, 928
 LIST_TO_CHARACTER_LINTG
 STRINGS, 654
 STRINGS::LIST_TO_CHARACTER, 928
 LIST_TO_CHARACTER_SP
 STRINGS, 655
 STRINGS::LIST_TO_CHARACTER, 929
 LIST_UNSORTED_TYPE
 LISTS_SortingOrder, 86
 LISTS, 460
 LIST_CREATE_FINISH, 464

LIST_CREATE_START, 464
LIST_DATA_TYPE_SET, 465
LIST_DESTROY, 465
LIST_DETACH_AND_DESTROY_DP, 466
LIST_DETACH_AND_DESTROY_INTG,
 466
LIST_DETACH_AND_DESTROY_SP, 467
LIST_FINALISE, 467
LIST_INITIAL_SIZE_SET, 468
LIST_INITIALISE, 468
LIST_ITEM_ADD_DP1, 468
LIST_ITEM_ADD_INTG1, 469
LIST_ITEM_ADD_SP1, 469
LIST_ITEM_DELETE, 470
LIST_ITEM_IN_LIST_DP1, 470
LIST_ITEM_IN_LIST_INTG1, 470
LIST_ITEM_IN_LIST_SP1, 471
LIST_NUMBER_OF_ITEMS_GET, 471
LIST_REMOVE_DUPLICATES, 472
LIST_SEARCH_DP_ARRAY, 472
LIST_SEARCH_INTG_ARRAY, 473
LIST_SEARCH_LINEAR_DP_ARRAY, 473
LIST_SEARCH_LINEAR_INTG_ARRAY,
 474
LIST_SEARCH_LINEAR_SP_ARRAY, 474
LIST_SEARCH_SP_ARRAY, 475
LIST_SORT_BUBBLE_DP_ARRAY, 475
LIST_SORT_BUBBLE_INTG_ARRAY, 475
LIST_SORT_BUBBLE_SP_ARRAY, 476
LIST_SORT_DP_ARRAY, 476
LIST_SORT_HEAP_DP_ARRAY, 476
LIST_SORT_HEAP_INTG_ARRAY, 477
LIST_SORT_HEAP_SP_ARRAY, 477
LIST_SORT_INTG_ARRAY, 477
LIST_SORT_SHELL_DP_ARRAY, 478
LIST_SORT_SHELL_INTG_ARRAY, 478
LIST_SORT_SHELL_SP_ARRAY, 478
LIST_SORT_SP_ARRAY, 479
LISTS::DataType, 84
LISTS::LIST_DETACH_AND_DESTROY, 849
 LIST_DETACH_AND_DESTROY_DP, 849
 LIST_DETACH_AND_DESTROY_INTG,
 849
 LIST_DETACH_AND_DESTROY_SP, 850
LISTS::LIST_ITEM_ADD, 851
 LIST_ITEM_ADD_DP1, 851
 LIST_ITEM_ADD_INTG1, 851
 LIST_ITEM_ADD_SP1, 851
LISTS::LIST_ITEM_IN_LIST, 853
 LIST_ITEM_IN_LIST_DP1, 853
 LIST_ITEM_IN_LIST_INTG1, 853
 LIST_ITEM_IN_LIST_SP1, 854
LISTS::LIST_PTR_TYPE, 855
 PTR, 855

LISTS::LIST_SEARCH, 856
 LIST_SEARCH_DP_ARRAY, 856
 LIST_SEARCH_INTG_ARRAY, 856
 LIST_SEARCH_SP_ARRAY, 857
LISTS::LIST_SEARCH_LINEAR, 858
 LIST_SEARCH_LINEAR_DP_ARRAY, 858
 LIST_SEARCH_LINEAR_INTG_ARRAY,
 858
 LIST_SEARCH_LINEAR_SP_ARRAY, 858
LISTS::LIST_SORT, 860
 LIST_SORT_DP_ARRAY, 860
 LIST_SORT_INTG_ARRAY, 860
 LIST_SORT_SP_ARRAY, 860
LISTS::LIST_SORT_BUBBLE, 862
 LIST_SORT_BUBBLE_DP_ARRAY, 862
 LIST_SORT_BUBBLE_INTG_ARRAY, 862
 LIST_SORT_BUBBLE_SP_ARRAY, 862
LISTS::LIST_SORT_HEAP, 864
 LIST_SORT_HEAP_DP_ARRAY, 864
 LIST_SORT_HEAP_INTG_ARRAY, 864
 LIST_SORT_HEAP_SP_ARRAY, 864
LISTS::LIST_SORT_SHELL, 866
 LIST_SORT_SHELL_DP_ARRAY, 866
 LIST_SORT_SHELL_INTG_ARRAY, 866
 LIST_SORT_SHELL_SP_ARRAY, 866
LISTS::LIST_TYPE, 867
 DATA_TYPE, 867
 INITIAL_SIZE, 868
 LIST_DP, 868
 LIST_FINISHED, 868
 LIST_INTG, 868
 LIST_SP, 868
 NUMBER_IN_LIST, 868
 SIZE, 868
 SORT_METHOD, 868
 SORT_ORDER, 869
LISTS::SortingMethod, 88
LISTS::SortingOrder, 86
LISTS_DataType
 LIST_DP_TYPE, 84
 LIST_INTG_TYPE, 84
 LIST_SP_TYPE, 85
LISTS_SortingOrder
 LIST_SORT_ASCENDING_TYPE, 86
 LIST_SORT_DESCENDING_TYPE, 86
 LIST_UNSORTED_TYPE, 86
LISTSSortingMethod
 LIST_BUBBLE_SORT_METHOD, 88
 LIST_HEAP_SORT_METHOD, 88
 LIST_SHELL_SORT_METHOD, 88
lle_CH_VS
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::lle, 833
lle_VS_CH

ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::lle, 833
 lle_VS_VS
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::lle, 833
 llt_CH_VS
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::llt, 834
 llt_VS_CH
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::llt, 834
 llt_VS_VS
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::llt, 834
 LOCAL_GHOST_RECEIVE_INDICES
 TYPES::DOMAIN_ADJACENT_-
 DOMAIN_TYPE, 993
 LOCAL_GHOST_SEND_INDICES
 TYPES::DOMAIN_ADJACENT_-
 DOMAIN_TYPE, 993
 LOCAL_LINE_XI_DIRECTION
 TYPES::BASIS_TYPE, 954
 LOCAL_NUMBER
 TYPES::DECOMPOSITION_ELEMENT_-
 TYPE, 962
 TYPES::DOMAIN_GLOBAL_MAPPING_-
 TYPE, 1005
 TYPES::DOMAIN_NODE_TYPE, 1018
 LOCAL_TO_GLOBAL_MAP
 TYPES::DOMAIN_MAPPING_TYPE, 1013
 LOCAL_TYPE
 TYPES::DOMAIN_GLOBAL_MAPPING_-
 TYPE, 1005
 logical
 binary_file_c.c, 1220
 f90c_c.c, 1272
 LOGICAL_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
 MACHINE_CONSTANTS, 481
 LOGICAL_TO_CHARACTER
 STRINGS, 655
 LOGICAL_TO_VSTRING
 STRINGS, 655
 LOGICALTYPE
 binary_file_c.c, 1219
 LONG_INTEGER_SIZE
 MACHINE_CONSTANTS, 481
 LONGINTTYPE
 binary_file_c.c, 1219

 M
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977

 TYPES::MATRIX_TYPE, 1131
 MACHINE_CHAR_FORMAT
 MACHINE_CONSTANTS, 481
 MACHINE_CONSTANTS, 480
 CHARACTER_SIZE, 480
 DOUBLE_COMPLEX_SIZE, 480
 DOUBLE_REAL_SIZE, 480
 ERROR_SEPARATOR_CONSTANT, 481
 INTEGER_SIZE, 481
 LOGICAL_SIZE, 481
 LONG_INTEGER_SIZE, 481
 MACHINE_CHAR_FORMAT, 481
 MACHINE_DP_FORMAT, 481
 MACHINE_ENDIAN, 481
 MACHINE_INT_FORMAT, 482
 MACHINE_OS, 482
 MACHINE_SP_FORMAT, 482
 MACHINE_TYPE, 482
 SHORT_INTEGER_SIZE, 482
 SINGLE_COMPLEX_SIZE, 482
 SINGLE_REAL_SIZE, 482
 MACHINE_DP_FORMAT
 MACHINE_CONSTANTS, 481
 MACHINE_ENDIAN
 MACHINE_CONSTANTS, 481
 MACHINE_INT_FORMAT
 MACHINE_CONSTANTS, 482
 MACHINE_OS
 MACHINE_CONSTANTS, 482
 MACHINE_SP_FORMAT
 MACHINE_CONSTANTS, 482
 MACHINE_TYPE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699
 MACHINE_CONSTANTS, 482
 MAPPINGS
 TYPES::DOMAIN_TYPE, 1025
 TYPES::FIELD_TYPE, 1110
 MatAssemblyBegin
 CMISS_PETSC::interface, 720
 MatAssemblyEnd
 CMISS_PETSC::interface, 720
 MatCreate
 CMISS_PETSC::interface, 720
 MatCreateMPIAIJ
 CMISS_PETSC::interface, 720
 MatCreateMPIDense
 CMISS_PETSC::interface, 720
 MatCreateSeqAIJ
 CMISS_PETSC::interface, 721
 MatCreateSeqDense
 CMISS_PETSC::interface, 721
 MatDestroy
 CMISS_PETSC::interface, 721

MATERIAL_FIELD
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
 TYPES::EQUATIONS_SET_MATERIALS_-
 TYPE, 1059
MATERIAL_INTERP_PARAMETERS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
MATERIAL_INTERP_POINT
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
MATERIALS
 TYPES::EQUATIONS_SET_TYPE, 1066
MATERIALS_FINISHED
 TYPES::EQUATIONS_SET_MATERIALS_-
 TYPE, 1059
MatGetArray
 CMISS_PETSC::interface, 721
MatGetArrayF90
 CMISS_PETSC::interface, 721
MatGetOwnershipRange
 CMISS_PETSC::interface, 721
MatGetValues
 CMISS_PETSC::interface, 721
MATHS, 484
 CROSS_PRODUCT_DP, 485
 CROSS_PRODUCT_INTG, 485
 CROSS_PRODUCT_SP, 485
 D_CROSS_PRODUCT_DP, 485
 D_CROSS_PRODUCT_INTG, 486
 D_CROSS_PRODUCT_SP, 486
 DETERMINANT_FULL_DP, 486
 DETERMINANT_FULL_INTG, 486
 DETERMINANT_FULL_SP, 487
 EDP_DP, 487
 EDP_SP, 487
 EIGENVALUE_FULL_DP, 487
 EIGENVALUE_FULL_SP, 487
 EIGENVECTOR_FULL_DP, 487
 EIGENVECTOR_FULL_SP, 488
 I0_DP, 488
 I0_SP, 488
 I1_DP, 488
 I1_SP, 488
 INVERT_FULL_DP, 488
 INVERT_FULL_SP, 489
 K0_DP, 489
 K0_SP, 489
 K1_DP, 489
 K1_SP, 489
 KDP_DP, 489
 KDP_SP, 489
 L2NORM_DP, 489
 L2NORM_SP, 490
NORMALISE_DP, 490
NORMALISE_SP, 490
SOLVE_SMALL_LINEAR_SYSTEM_DP,
 490
SOLVE_SMALL_LINEAR_SYSTEM_SP,
 490
MATHS::CROSS_PRODUCT, 870
 CROSS_PRODUCT_DP, 870
 CROSS_PRODUCT_INTG, 870
 CROSS_PRODUCT_SP, 870
MATHS::D_CROSS_PRODUCT, 871
 D_CROSS_PRODUCT_DP, 871
 D_CROSS_PRODUCT_INTG, 871
 D_CROSS_PRODUCT_SP, 871
MATHS::DETERMINANT, 872
 DETERMINANT_FULL_DP, 872
 DETERMINANT_FULL_INTG, 872
 DETERMINANT_FULL_SP, 872
MATHS::EDP, 873
 EDP_DP, 873
 EDP_SP, 873
MATHS::EIGENVALUE, 874
 EIGENVALUE_FULL_DP, 874
 EIGENVALUE_FULL_SP, 874
MATHS::EIGENVECTOR, 875
 EIGENVECTOR_FULL_DP, 875
 EIGENVECTOR_FULL_SP, 875
MATHS::I0, 876
 I0_DP, 876
 I0_SP, 876
MATHS::I1, 877
 I1_DP, 877
 I1_SP, 877
MATHS::INVERT, 878
 INVERT_FULL_DP, 878
 INVERT_FULL_SP, 878
MATHS::K0, 879
 K0_DP, 879
 K0_SP, 879
MATHS::K1, 880
 K1_DP, 880
 K1_SP, 880
MATHS::L2NORM, 881
 L2NORM_DP, 881
 L2NORM_SP, 881
MATHS::NORMALISE, 882
 NORMALISE_DP, 882
 NORMALISE_SP, 882
MATHS::SOLVE_SMALL_LINEAR_SYSTEM,
 883
 SOLVE_SMALL_LINEAR_SYSTEM_DP,
 883
 SOLVE_SMALL_LINEAR_SYSTEM_SP,
 883

MatRestoreArray
 CMISS_PETSC::interface, 721

MatRestoreArrayF90
 CMISS_PETSC::interface, 721

MATRICES
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044

MATRIX
 TYPES::DISTRIBUTED_MATRIX_-
 CMISS_TYPE, 974

MATRIX_ALL_VALUES_SET_DP
 MATRIX_VECTOR, 499

MATRIX_ALL_VALUES_SET_INTG
 MATRIX_VECTOR, 499

MATRIX_ALL_VALUES_SET_L
 MATRIX_VECTOR, 500

MATRIX_ALL_VALUES_SET_SP
 MATRIX_VECTOR, 500

MATRIX_BLOCK_STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 93

MATRIX_COEFFICIENT
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048

MATRIX_COEFFICIENTS
 TYPES::EQUATIONS_MAPPING_-
 CREATE_VALUES_CACHE_TYPE,
 1038

MATRIX_COLUMN_MAJOR_STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 94

MATRIX_COMPRESSED_COLUMN_-
 STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 94

MATRIX_COMPRESSED_ROW_STORAGE_-
 TYPE
 MATRIX_VECTOR_StorageTypes, 94

MATRIX_CREATE_FINISH
 MATRIX_VECTOR, 500

MATRIX_CREATE_START
 MATRIX_VECTOR, 501

MATRIX_DATA_GET_DP
 MATRIX_VECTOR, 501

MATRIX_VECTOR::MATRIX_DATA_GET,
 886

MATRIX_DATA_GET_INTG
 MATRIX_VECTOR, 502

MATRIX_VECTOR::MATRIX_DATA_GET,
 886

MATRIX_DATA_GET_L
 MATRIX_VECTOR, 502

MATRIX_VECTOR::MATRIX_DATA_GET,
 886

MATRIX_DATA_GET_SP
 MATRIX_VECTOR, 502

MATRIX_VECTOR::MATRIX_DATA_GET,
 887

MATRIX_DATA_TYPE_SET
 MATRIX_VECTOR, 503

MATRIX_DESTROY
 MATRIX_VECTOR, 503

MATRIX_DIAGONAL_STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 95

MATRIX_DUPLICATE
 MATRIX_VECTOR, 504

MATRIX_FINALISE
 MATRIX_VECTOR, 504

MATRIX_FINISHED
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 980

MATRIX_INITIALISE
 MATRIX_VECTOR, 505

MATRIX_MAX_COLUMNS_PER_ROW_GET
 MATRIX_VECTOR, 505

MATRIX_MAX_SIZE_SET
 MATRIX_VECTOR, 505

MATRIX_NUMBER
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048

MATRIX_NUMBER_NON_ZEROS_SET
 MATRIX_VECTOR, 506

MATRIX_OUTPUT
 MATRIX_VECTOR, 506

MATRIX_ROW_COLUMN_STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 95

MATRIX_ROW_MAJOR_STORAGE_TYPE
 MATRIX_VECTOR_StorageTypes, 95

MATRIX_SIZE_SET
 MATRIX_VECTOR, 507

MATRIX_STORAGE_LOCATION_FIND
 MATRIX_VECTOR, 507

MATRIX_STORAGE_LOCATIONS_GET
 MATRIX_VECTOR, 508

MATRIX_STORAGE_LOCATIONS_SET
 MATRIX_VECTOR, 509

MATRIX_STORAGE_TYPE_GET
 MATRIX_VECTOR, 509

MATRIX_STORAGE_TYPE_SET
 MATRIX_VECTOR, 510

MATRIX_VALUES_ADD_DP
 MATRIX_VECTOR, 510

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 888

MATRIX_VALUES_ADD_DP1
 MATRIX_VECTOR, 511

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 889

MATRIX_VALUES_ADD_DP2
 MATRIX_VECTOR, 511

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 889

MATRIX_VALUES_ADD_INTG
 MATRIX_VECTOR, 512

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 889

MATRIX_VALUES_ADD_INTG1
 MATRIX_VECTOR, 512

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 890

MATRIX_VALUES_ADD_INTG2
 MATRIX_VECTOR, 513

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 890

MATRIX_VALUES_ADD_L
 MATRIX_VECTOR, 513

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 890

MATRIX_VALUES_ADD_L1
 MATRIX_VECTOR, 514

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 891

MATRIX_VALUES_ADD_L2
 MATRIX_VECTOR, 514

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 891

MATRIX_VALUES_ADD_SP
 MATRIX_VECTOR, 515

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 891

MATRIX_VALUES_ADD_SP1
 MATRIX_VECTOR, 515

 MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 892

MATRIX_VALUES_ADD_SP2
 MATRIX_VECTOR, 516

MATRIX_VECTOR::MATRIX_VALUES_-
 ADD, 892

MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 893

MATRIX_VALUES_GET_DP1
 MATRIX_VECTOR, 517

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 894

MATRIX_VALUES_GET_DP2
 MATRIX_VECTOR, 517

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 894

MATRIX_VALUES_GET_INTG
 MATRIX_VECTOR, 518

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 894

MATRIX_VALUES_GET_INTG1
 MATRIX_VECTOR, 518

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 895

MATRIX_VALUES_GET_INTG2
 MATRIX_VECTOR, 519

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 895

MATRIX_VALUES_GET_L
 MATRIX_VECTOR, 519

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 895

MATRIX_VALUES_GET_L1
 MATRIX_VECTOR, 520

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 896

MATRIX_VALUES_GET_L2
 MATRIX_VECTOR, 520

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 896

MATRIX_VALUES_GET_SP
 MATRIX_VECTOR, 521

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 896

MATRIX_VALUES_GET_SP1
 MATRIX_VECTOR, 521

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 897

MATRIX_VALUES_GET_SP2
 MATRIX_VECTOR, 522

 MATRIX_VECTOR::MATRIX_VALUES_-
 GET, 897

MATRIX_VALUES_SET_DP
 MATRIX_VECTOR, 522

 MATRIX_VECTOR::MATRIX_VALUES_-
 SET, 898

MATRIX_VALUES_SET_DP1
 MATRIX_VECTOR, 523
 MATRIX_VECTOR::MATRIX_VALUES_SET, 899

MATRIX_VALUES_SET_DP2
 MATRIX_VECTOR, 523
 MATRIX_VECTOR::MATRIX_VALUES_SET, 899

MATRIX_VALUES_SET_INTG
 MATRIX_VECTOR, 524
 MATRIX_VECTOR::MATRIX_VALUES_SET, 899

MATRIX_VALUES_SET_INTG1
 MATRIX_VECTOR, 524
 MATRIX_VECTOR::MATRIX_VALUES_SET, 900

MATRIX_VALUES_SET_INTG2
 MATRIX_VECTOR, 525
 MATRIX_VECTOR::MATRIX_VALUES_SET, 900

MATRIX_VALUES_SET_L
 MATRIX_VECTOR, 525
 MATRIX_VECTOR::MATRIX_VALUES_SET, 900

MATRIX_VALUES_SET_L1
 MATRIX_VECTOR, 526
 MATRIX_VECTOR::MATRIX_VALUES_SET, 901

MATRIX_VALUES_SET_L2
 MATRIX_VECTOR, 526
 MATRIX_VECTOR::MATRIX_VALUES_SET, 901

MATRIX_VALUES_SET_SP
 MATRIX_VECTOR, 527
 MATRIX_VECTOR::MATRIX_VALUES_SET, 901

MATRIX_VALUES_SET_SP1
 MATRIX_VECTOR, 527
 MATRIX_VECTOR::MATRIX_VALUES_SET, 902

MATRIX_VALUES_SET_SP2
 MATRIX_VECTOR, 528
 MATRIX_VECTOR::MATRIX_VALUES_SET, 902

MATRIX_VARIABLE_TYPES
 TYPES::EQUATIONS_MAPPING_CREATE_VALUES_CACHE_TYPE, 1038
 TYPES::EQUATIONS_MAPPING_TYPE, 1041
 TYPES::SOLUTION_MAPPING_CREATE_VALUES_CACHE_TYPE, 1173

MATRIX_VECTOR, 491
 MATRIX_ALL_VALUES_SET_DP, 499

MATRIX_ALL_VALUES_SET_INTG, 499
 MATRIX_ALL_VALUES_SET_L, 500
 MATRIX_ALL_VALUES_SET_SP, 500

MATRIX_CREATE_FINISH, 500
 MATRIX_CREATE_START, 501
 MATRIX_DATA_GET_DP, 501
 MATRIX_DATA_GET_INTG, 502
 MATRIX_DATA_GET_L, 502
 MATRIX_DATA_GET_SP, 502

MATRIX_DATA_TYPE_SET, 503
 MATRIX_DESTROY, 503
 MATRIX_DUPLICATE, 504
 MATRIX_FINALISE, 504
 MATRIX_INITIALISE, 505
 MATRIX_MAX_COLUMNS_PER_ROW_GET, 505

MATRIX_MAX_SIZE_SET, 505
 MATRIX_NUMBER_NON_ZEROS_SET, 506

MATRIX_OUTPUT, 506
 MATRIX_SIZE_SET, 507

MATRIX_STORAGE_LOCATION_FIND, 507

MATRIX_STORAGE_LOCATIONS_GET, 508

MATRIX_STORAGE_LOCATIONS_SET, 509

MATRIX_STORAGE_TYPE_GET, 509
 MATRIX_STORAGE_TYPE_SET, 510

MATRIX_VALUES_ADD_DP, 510
 MATRIX_VALUES_ADD_DP1, 511
 MATRIX_VALUES_ADD_DP2, 511
 MATRIX_VALUES_ADD_INTG, 512
 MATRIX_VALUES_ADD_INTG1, 512
 MATRIX_VALUES_ADD_INTG2, 513
 MATRIX_VALUES_ADD_L, 513
 MATRIX_VALUES_ADD_L1, 514
 MATRIX_VALUES_ADD_L2, 514
 MATRIX_VALUES_ADD_SP, 515
 MATRIX_VALUES_ADD_SP1, 515
 MATRIX_VALUES_ADD_SP2, 516
 MATRIX_VALUES_GET_DP, 516
 MATRIX_VALUES_GET_DP1, 517
 MATRIX_VALUES_GET_DP2, 517
 MATRIX_VALUES_GET_INTG, 518
 MATRIX_VALUES_GET_INTG1, 518
 MATRIX_VALUES_GET_INTG2, 519
 MATRIX_VALUES_GET_L, 519
 MATRIX_VALUES_GET_L1, 520
 MATRIX_VALUES_GET_L2, 520
 MATRIX_VALUES_GET_SP, 521
 MATRIX_VALUES_GET_SP1, 521
 MATRIX_VALUES_GET_SP2, 522

MATRIX_VALUES_SET_DP, 522

MATRIX_VALUES_SET_DP1, 523
MATRIX_VALUES_SET_DP2, 523
MATRIX_VALUES_SET_INTG, 524
MATRIX_VALUES_SET_INTG1, 524
MATRIX_VALUES_SET_INTG2, 525
MATRIX_VALUES_SET_L, 525
MATRIX_VALUES_SET_L1, 526
MATRIX_VALUES_SET_L2, 526
MATRIX_VALUES_SET_SP, 527
MATRIX_VALUES_SET_SP1, 527
MATRIX_VALUES_SET_SP2, 528
MATRIX_VECTOR_ID, 541
VECTOR_ALL_VALUES_SET_DP, 528
VECTOR_ALL_VALUES_SET_INTG, 528
VECTOR_ALL_VALUES_SET_L, 529
VECTOR_ALL_VALUES_SET_SP, 529
VECTOR_CREATE_FINISH, 530
VECTOR_CREATE_START, 530
VECTOR_DATA_GET_DP, 530
VECTOR_DATA_GET_INTG, 531
VECTOR_DATA_GET_L, 531
VECTOR_DATA_GET_SP, 532
VECTOR_DATA_TYPE_SET, 532
VECTOR_DESTROY, 533
VECTOR_DUPLICATE, 533
VECTOR_FINALISE, 533
VECTOR_INITIALISE, 534
VECTOR_SIZE_SET, 534
VECTOR_VALUES_GET_DP, 535
VECTOR_VALUES_GET_DP1, 535
VECTOR_VALUES_GET_INTG, 535
VECTOR_VALUES_GET_INTG1, 536
VECTOR_VALUES_GET_L, 536
VECTOR_VALUES_GET_L1, 537
VECTOR_VALUES_GET_SP, 537
VECTOR_VALUES_GET_SP1, 537
VECTOR_VALUES_SET_DP, 538
VECTOR_VALUES_SET_DP1, 538
VECTOR_VALUES_SET_INTG, 539
VECTOR_VALUES_SET_INTG1, 539
VECTOR_VALUES_SET_L, 539
VECTOR_VALUES_SET_L1, 540
VECTOR_VALUES_SET_SP, 540
VECTOR_VALUES_SET_SP1, 541
MATRIX_VECTOR::DataTypes, 90
MATRIX_VECTOR::MATRIX_ALL_VALUES_SET, 884
MATRIX_ALL_VALUES_SET_DP, 884
MATRIX_ALL_VALUES_SET_INTG, 884
MATRIX_ALL_VALUES_SET_L, 884
MATRIX_ALL_VALUES_SET_SP, 885
MATRIX_VECTOR::MATRIX_DATA_GET, 886
MATRIX_DATA_GET_DP, 886
MATRIX_DATA_GET_INTG, 886
MATRIX_DATA_GET_L, 886
MATRIX_DATA_GET_SP1, 886
MATRIX_VECTOR::MATRIX_VALUES_ADD, 888
MATRIX_VALUES_ADD_DP, 888
MATRIX_VALUES_ADD_DP1, 889
MATRIX_VALUES_ADD_DP2, 889
MATRIX_VALUES_ADD_INTG, 889
MATRIX_VALUES_ADD_INTG1, 890
MATRIX_VALUES_ADD_INTG2, 890
MATRIX_VALUES_ADD_L, 890
MATRIX_VALUES_ADD_L1, 891
MATRIX_VALUES_ADD_L2, 891
MATRIX_VALUES_ADD_SP, 891
MATRIX_VALUES_ADD_SP1, 892
MATRIX_VALUES_ADD_SP2, 892
MATRIX_VECTOR::MATRIX_VALUES_GET, 893
MATRIX_VALUES_GET_DP, 893
MATRIX_VALUES_GET_DP1, 894
MATRIX_VALUES_GET_DP2, 894
MATRIX_VALUES_GET_INTG, 894
MATRIX_VALUES_GET_INTG1, 895
MATRIX_VALUES_GET_INTG2, 895
MATRIX_VALUES_GET_L, 895
MATRIX_VALUES_GET_L1, 896
MATRIX_VALUES_GET_L2, 896
MATRIX_VALUES_GET_SP, 896
MATRIX_VALUES_GET_SP1, 897
MATRIX_VALUES_GET_SP2, 897
MATRIX_VECTOR::MATRIX_VALUES_SET, 898
MATRIX_VALUES_SET_DP, 898
MATRIX_VALUES_SET_DP1, 899
MATRIX_VALUES_SET_DP2, 899
MATRIX_VALUES_SET_INTG, 899
MATRIX_VALUES_SET_INTG1, 900
MATRIX_VALUES_SET_INTG2, 900
MATRIX_VALUES_SET_L, 900
MATRIX_VALUES_SET_L1, 901
MATRIX_VALUES_SET_L2, 901
MATRIX_VALUES_SET_SP, 901
MATRIX_VALUES_SET_SP1, 902
MATRIX_VALUES_SET_SP2, 902
MATRIX_VECTOR::StorageTypes, 93
MATRIX_VECTOR::VECTOR_ALL_VALUES_SET, 903
VECTOR_ALL_VALUES_SET_DP, 903
VECTOR_ALL_VALUES_SET_INTG, 903
VECTOR_ALL_VALUES_SET_L, 903
VECTOR_ALL_VALUES_SET_SP, 904
MATRIX_VECTOR::VECTOR_DATA_GET, 905
VECTOR_DATA_GET_DP, 905
VECTOR_DATA_GET_INTG, 905

VECTOR_DATA_GET_L, 905
 VECTOR_DATA_GET_SP, 906
 MATRIX_VECTOR::VECTOR_VALUES_GET,
 907
 VECTOR_VALUES_GET_DP, 907
 VECTOR_VALUES_GET_DP1, 907
 VECTOR_VALUES_GET_INTG, 907
 VECTOR_VALUES_GET_INTG1, 908
 VECTOR_VALUES_GET_L, 908
 VECTOR_VALUES_GET_L1, 908
 VECTOR_VALUES_GET_SP, 909
 VECTOR_VALUES_GET_SP1, 909
 MATRIX_VECTOR::VECTOR_VALUES_SET,
 910
 VECTOR_VALUES_SET_DP, 910
 VECTOR_VALUES_SET_DP1, 910
 VECTOR_VALUES_SET_INTG, 910
 VECTOR_VALUES_SET_INTG1, 911
 VECTOR_VALUES_SET_L, 911
 VECTOR_VALUES_SET_L1, 911
 VECTOR_VALUES_SET_SP, 912
 VECTOR_VALUES_SET_SP1, 912
 MATRIX_VECTOR_DataTypes
 MATRIX_VECTOR_DP_TYPE, 90
 MATRIX_VECTOR_INTG_TYPE, 91
 MATRIX_VECTOR_L_TYPE, 91
 MATRIX_VECTOR_SP_TYPE, 92
 MATRIX_VECTOR_DP_TYPE
 MATRIX_VECTOR_DataTypes, 90
 MATRIX_VECTOR_ID
 MATRIX_VECTOR, 541
 MATRIX_VECTOR_INTG_TYPE
 MATRIX_VECTOR_DataTypes, 91
 MATRIX_VECTOR_L_TYPE
 MATRIX_VECTOR_DataTypes, 91
 MATRIX_VECTOR_SP_TYPE
 MATRIX_VECTOR_DataTypes, 92
 MATRIX_VECTOR_StorageTypes
 MATRIX_BLOCK_STORAGE_TYPE, 93
 MATRIX_COLUMN_MAJOR_STORAGE_-
 TYPE, 94
 MATRIX_COMPRESSED_COLUMN_-
 STORAGE_TYPE, 94
 MATRIX_COMPRESSED_ROW_-
 STORAGE_TYPE, 94
 MATRIX_DIAGONAL_STORAGE_TYPE,
 95
 MATRIX_ROW_COLUMN_STORAGE_-
 TYPE, 95
 MATRIX_ROW_MAJOR_STORAGE_-
 TYPE, 95
 MatSetLocalToGlobalMapping
 CMISS_PETSC::interface, 721
 MatSetOption

CMISS_PETSC::interface, 722
 MatSetSizes
 CMISS_PETSC::interface, 722
 MatSetValue
 CMISS_PETSC::interface, 722
 MatSetValueLocal
 CMISS_PETSC::interface, 722
 MatSetValues
 CMISS_PETSC::interface, 722
 MatSetValuesLocal
 CMISS_PETSC::interface, 722
 MatView
 CMISS_PETSC::interface, 722
 MatZeroEntries
 CMISS_PETSC::interface, 722
 MAX_M
 TYPES::MATRIX_TYPE, 1131
 MAX_N
 TYPES::MATRIX_TYPE, 1131
 MAX_NUM_BINARY_FILES
 BINARY_FILE, 189
 MAX_NUMBER_OF_COLUMNS
 TYPES::ELEMENT_MATRIX_TYPE, 1028
 MAX_NUMBER_OF_DERIVATIVES
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097
 TYPES::FIELD_SCALING_TYPE, 1105
 MAX_NUMBER_OF_ELEMENT_-
 PARAMETERS
 TYPES::FIELD_SCALING_TYPE, 1105
 MAX_NUMBER_OF_INTERPOLATION_-
 PARAMETERS
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113
 TYPES::FIELD_VARIABLE_TYPE, 1117
 MAX_NUMBER_OF_ROWS
 TYPES::ELEMENT_MATRIX_TYPE, 1028
 TYPES::ELEMENT_VECTOR_TYPE, 1030
 MAX_OUTPUT_LINES
 BASE_ROUTINES, 173
 MAX_PARTIAL_DERIVATIVE_INDEX
 TYPES::FIELD_INTERPOLATED_POINT_-
 TYPE, 1091
 MAXBINFILES
 binary_file_c.c, 1219
 MAXIMUM_COLUMN_INDICES_PER_ROW
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977
 TYPES::MATRIX_TYPE, 1131
 MAXIMUM_EXTENT
 TYPES::GENERATED_MESH_-
 REGULAR_TYPE, 1120
 MAXIMUM_NUMBER_OF_DERIVATIVES
 TYPES::BASIS_TYPE, 954

TYPES::DOMAIN_NODES_TYPE, 1020
MAXIMUM_NUMBER_OF_ELEMENT_-
PARAMETERS
 TYPES::DOMAIN_ELEMENTS_TYPE, 999
MAXIMUM_NUMBER_OF_ITERATIONS
 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1125
MESH
 TYPES::DECOMPOSITION_TYPE, 972
 TYPES::DECOMPOSITIONS_TYPE, 973
 TYPES::DOMAIN_TYPE, 1026
 TYPES::GENERATED_MESH_TYPE, 1122
 TYPES::MESH_DOFS_TYPE, 1133
 TYPES::MESH_ELEMENTS_TYPE, 1136
 TYPES::MESH_NODES_TYPE, 1140
 TYPES::MESH_TOPOLOGY_TYPE, 1143
MESH_COMPONENT_NUMBER
 TYPES::DECOMPOSITION_TYPE, 972
 TYPES::DOMAIN_TYPE, 1026
 TYPES::FIELD_CREATE_VALUES_-
 CACHE_TYPE, 1080
 TYPES::FIELD_SCALING_TYPE, 1105
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113
 TYPES::MESH_TOPOLOGY_TYPE, 1143
MESH_CREATE_FINISH
 MESH_ROUTINES, 560
MESH_CREATE_REGULAR
 MESH_ROUTINES, 560
MESH_CREATE_START
 MESH_ROUTINES, 561
MESH_DESTROY
 MESH_ROUTINES, 562
MESH_DIMENSION
 TYPES::GENERATED_MESH_-
 REGULAR_TYPE, 1121
MESH_EMBEDDED
 TYPES::MESH_TYPE, 1146
MESH_FINALISE
 MESH_ROUTINES, 562
MESH_FINISHED
 TYPES::MESH_TYPE, 1146
MESH_INITIALISE
 MESH_ROUTINES, 563
MESH_NUMBER_OF_COMPONENTS_SET_-
 NUMBER
 MESH_ROUTINES, 563
 MESH_ROUTINES::MESH_NUMBER_-
 OF_COMPONENTS_SET, 913
MESH_NUMBER_OF_COMPONENTS_SET_-
 PTR
 MESH_ROUTINES, 564
 MESH_ROUTINES::MESH_NUMBER_-
 OF_COMPONENTS_SET, 913
MESH_NUMBER_OF_ELEMENTS_SET_-
 NUMBER
 MESH_ROUTINES, 564
 MESH_ROUTINES::MESH_NUMBER_-
 OF_ELEMENTS_SET, 914
MESH_NUMBER_OF_ELEMENTS_SET_PTR
 MESH_ROUTINES, 564
 MESH_ROUTINES::MESH_NUMBER_-
 OF_ELEMENTS_SET, 914
MESH_ROUTINES, 542
 DECOMPOSITION_CREATE_FINISH, 547
 DECOMPOSITION_CREATE_START, 547
 DECOMPOSITION_DESTROY, 548
 DECOMPOSITION_ELEMENT_-
 DOMAIN_CALCULATE, 548
 DECOMPOSITION_ELEMENT_-
 DOMAIN_SET, 549
 DECOMPOSITION_MESH_-
 COMPONENT_NUMBER_SET, 549
 DECOMPOSITION_NUMBER_OF_-
 DOMAINS_SET, 550
 DOMAIN_MAPPINGS_NODES_FINALISE,
 550
 DOMAIN_MAPPINGS_NODES_-
 INITIALISE, 551
 DOMAIN_TOPOLOGY_CALCULATE, 551
 DOMAIN_TOPOLOGY_DOFS_FINALISE,
 552
 DOMAIN_TOPOLOGY_DOFS_-
 INITIALISE, 552
 DOMAIN_TOPOLOGY_ELEMENT_-
 FINALISE, 553
 DOMAIN_TOPOLOGY_ELEMENT_-
 INITIALISE, 553
 DOMAIN_TOPOLOGY_ELEMENTS_-
 FINALISE, 553
 DOMAIN_TOPOLOGY_ELEMENTS_-
 INITIALISE, 554
 DOMAIN_TOPOLOGY_FINALISE, 554
 DOMAIN_TOPOLOGY_INITIALISE, 555
 DOMAIN_TOPOLOGY_INITIALISE_-
 FROM_MESH, 555
 DOMAIN_TOPOLOGY_LINE_FINALISE,
 556
 DOMAIN_TOPOLOGY_LINE_INITIALISE,
 556
 DOMAIN_TOPOLOGY_LINES_FINALISE,
 557
 DOMAIN_TOPOLOGY_LINES_-
 INITIALISE, 557
 DOMAIN_TOPOLOGY_NODE_FINALISE,
 558
 DOMAIN_TOPOLOGY_NODE_-
 INITIALISE, 558

DOMAIN_TOPOLOGY_NODES_-
 FINALISE, 558
 DOMAIN_TOPOLOGY_NODES_-
 INITIALISE, 559
 DOMAIN_TOPOLOGY_NODES_-
 SURROUNDING_ELEMENTS_-
 CALCULATE, 559
 MESH_CREATE_FINISH, 560
 MESH_CREATE_REGULAR, 560
 MESH_CREATE_START, 561
 MESH_DESTROY, 562
 MESH_FINALISE, 562
 MESH_INITIALISE, 563
 MESH_NUMBER_OF_COMPONENTS_-
 SET_NUMBER, 563
 MESH_NUMBER_OF_COMPONENTS_-
 SET_PTR, 564
 MESH_NUMBER_OF_ELEMENTS_SET_-
 NUMBER, 564
 MESH_NUMBER_OF_ELEMENTS_SET_-
 PTR, 564
 MESH_TOPOLOGY_CALCULATE, 565
 MESH_TOPOLOGY_DOFS_CALCULATE,
 565
 MESH_TOPOLOGY_DOFS_FINALISE, 566
 MESH_TOPOLOGY_DOFS_INITIALISE,
 566
 MESH_TOPOLOGY_ELEMENT_-
 FINALISE, 567
 MESH_TOPOLOGY_ELEMENT_-
 INITIALISE, 567
 MESH_TOPOLOGY_ELEMENTS_-
 ADJACENT_ELEMENTS_-
 CALCULATE, 568
 MESH_TOPOLOGY_ELEMENTS_BASIS_-
 SET, 568
 MESH_TOPOLOGY_ELEMENTS_-
 CREATE_FINISH, 569
 MESH_TOPOLOGY_ELEMENTS_-
 CREATE_START, 569
 MESH_TOPOLOGY_ELEMENTS_-
 DESTROY, 570
 MESH_TOPOLOGY_ELEMENTS_-
 ELEMENT_BASIS_SET, 570
 MESH_TOPOLOGY_ELEMENTS_-
 ELEMENT_NODES_SET, 571
 MESH_TOPOLOGY_ELEMENTS_-
 FINALISE, 571
 MESH_TOPOLOGY_ELEMENTS_-
 INITIALISE, 572
 MESH_TOPOLOGY_ELEMENTS_-
 NUMBER_SET, 572
 MESH_TOPOLOGY_FINALISE, 573
 MESH_TOPOLOGY_INITIALISE, 573
 MESH_TOPOLOGY_NODE_FINALISE,
 574
 MESH_TOPOLOGY_NODE_INITIALISE,
 574
 MESH_TOPOLOGY_NODES_-
 CALCULATE, 575
 MESH_TOPOLOGY_NODES_-
 DERIVATIVES_CALCULATE, 575
 MESH_TOPOLOGY_NODES_FINALISE,
 576
 MESH_TOPOLOGY_NODES_INITIALISE,
 576
 MESH_TOPOLOGY_NODES_-
 SURROUNDING_ELEMENTS_-
 CALCULATE, 576
 MESH_USER_NUMBER_FIND, 577
 MESSES_FINALISE, 577
 MESSES_INITIALISE, 578
 MESH_ROUTINES::DecompositionTypes, 97
 MESH_ROUTINES::MESH_NUMBER_OF_-
 COMPONENTS_SET, 913
 MESH_NUMBER_OF_COMPONENTS_-
 SET_NUMBER, 913
 MESH_NUMBER_OF_COMPONENTS_-
 SET_PTR, 913
 MESH_ROUTINES::MESH_NUMBER_OF_-
 ELEMENTS_SET, 914
 MESH_NUMBER_OF_ELEMENTS_SET_-
 NUMBER, 914
 MESH_NUMBER_OF_ELEMENTS_SET_-
 PTR, 914
 MESH_ROUTINES_DecompositionTypes
 DECOMPOSITION_ALL_TYPE, 97
 DECOMPOSITION_CALCULATED_TYPE,
 97
 DECOMPOSITION_USER_DEFINED_-
 TYPE, 98
 MESH_TOPOLOGY_CALCULATE
 MESH_ROUTINES, 565
 MESH_TOPOLOGY_DOFS_CALCULATE
 MESH_ROUTINES, 565
 MESH_TOPOLOGY_DOFS_FINALISE
 MESH_ROUTINES, 566
 MESH_TOPOLOGY_DOFS_INITIALISE
 MESH_ROUTINES, 566
 MESH_TOPOLOGY_ELEMENT_FINALISE
 MESH_ROUTINES, 567
 MESH_TOPOLOGY_ELEMENT_INITIALISE
 MESH_ROUTINES, 567
 MESH_TOPOLOGY_ELEMENTS_-
 ADJACENT_ELEMENTS_-
 CALCULATE
 MESH_ROUTINES, 568
 MESH_TOPOLOGY_ELEMENTS_BASIS_SET

MESH_ROUTINES, 568
MESH_TOPOLOGY_ELEMENTS_CREATE_-
 FINISH
 MESH_ROUTINES, 569
MESH_TOPOLOGY_ELEMENTS_CREATE_-
 START
 MESH_ROUTINES, 569
MESH_TOPOLOGY_ELEMENTS_DESTROY
 MESH_ROUTINES, 570
MESH_TOPOLOGY_ELEMENTS_ELEMENT_-
 BASIS_SET
 MESH_ROUTINES, 570
MESH_TOPOLOGY_ELEMENTS_ELEMENT_-
 NODES_SET
 MESH_ROUTINES, 571
MESH_TOPOLOGY_ELEMENTS_FINALISE
 MESH_ROUTINES, 571
MESH_TOPOLOGY_ELEMENTS_INITIALISE
 MESH_ROUTINES, 572
MESH_TOPOLOGY_ELEMENTS_NUMBER_-
 SET
 MESH_ROUTINES, 572
MESH_TOPOLOGY_FINALISE
 MESH_ROUTINES, 573
MESH_TOPOLOGY_INITIALISE
 MESH_ROUTINES, 573
MESH_TOPOLOGY_NODE_FINALISE
 MESH_ROUTINES, 574
MESH_TOPOLOGY_NODE_INITIALISE
 MESH_ROUTINES, 574
MESH_TOPOLOGY_NODES_CALCULATE
 MESH_ROUTINES, 575
MESH_TOPOLOGY_NODES_DERIVATIVES_-
 CALCULATE
 MESH_ROUTINES, 575
MESH_TOPOLOGY_NODES_FINALISE
 MESH_ROUTINES, 576
MESH_TOPOLOGY_NODES_INITIALISE
 MESH_ROUTINES, 576
MESH_TOPOLOGY_NODES_-
 SURROUNDING_ELEMENTS_-
 CALCULATE
 MESH_ROUTINES, 576
MESH_USER_NUMBER_FIND
 MESH_ROUTINES, 577
MESHES
 TYPES::MESH_TYPE, 1147
 TYPES::MESHES_TYPE, 1149
 TYPES::REGION_TYPE, 1171
MESHES_FINALISE
 MESH_ROUTINES, 577
MESHES_INITIALISE
 MESH_ROUTINES, 578
MPI_COMM
COMP_ENVIRONMENT::COMPUTATIONAL_-
 ENVIRONMENT_TYPE, 735
MPI_COMPUTATIONAL_NODE_TYPE_DATA
 COMP_ENVIRONMENT, 246
MPI_ERROR_CHECK
 CMISS_MPI, 197
MPI_RECEIVE_REQUEST
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 988
MPI_SEND_REQUEST
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 988
MPI_TYPE
 COMP_ENVIRONMENT::MPI_-
 COMPUTATIONAL_NODE_TYPE,
 739
MY_COMPUTATIONAL_NODE_NUMBER
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 ENVIRONMENT_TYPE, 735

N

TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 977
TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, 983
TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, 986
TYPES::MATRIX_TYPE, 1131
TYPES::VECTOR_TYPE, 1207

NAME

BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 691
BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695

NEXT_ROUTINE

BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 691

NIL

TREES::TREE_TYPE, 944

NODAL_INFO_SET

FIELD_IO_ROUTINES::FIELD_IO_-
 NODAL_INFO_SET, 764

NODE_AT_COLLAPSE

 TYPES::BASIS_TYPE, 954

NODE_CHECK_EXISTS

 NODE_ROUTINES, 579

NODE_DESTROY

 NODE_ROUTINES, 579

NODE_DOF2PARAM_MAP

 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1083

NODE_DOMAIN

 TYPES::DOMAIN_TYPE, 1026

NODE_INITIAL_POSITION_SET

NODE_ROUTINES, 579
 NODE_LINES
 TYPES::DOMAIN_NODE_TYPE, 1018
 NODE_NAME
 COMP_ENVIRONMENT::COMPUTATIONAL_NORMALISE_DP
 NODE_TYPE, 737
 NODE_NAME_LENGTH
 COMP_ENVIRONMENT::COMPUTATIONAL_NORMALISE_SP
 NODE_TYPE, 737
 NODE_NUMBER_SET
 NODE_ROUTINES, 580
 NODE_NUMBERS_IN_LOCAL_LINE
 TYPES::BASIS_TYPE, 954
 NODE_PARAM2DOF_MAP
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097
 NODE_POSITION_INDEX
 TYPES::BASIS_TYPE, 954
 NODE_POSITION_INDEX_INV
 TYPES::BASIS_TYPE, 954
 NODE_ROUTINES, 579
 NODE_CHECK_EXISTS, 579
 NODE_DESTROY, 579
 NODE_INITIAL_POSITION_SET, 579
 NODE_NUMBER_SET, 580
 NODES_CREATE_FINISH, 580
 NODES_CREATE_START, 580
 NODES_FINALISE, 581
 NODES_INITIALISE, 581
 NODE_TREE
 TYPES::NODES_TYPE, 1152
 NODES
 TYPES::DOMAIN_MAPPINGS_TYPE, 1015
 TYPES::DOMAIN_NODES_TYPE, 1020
 TYPES::DOMAIN_TOPOLOGY_TYPE,
 1024
 TYPES::MESH_NODES_TYPE, 1140
 TYPES::MESH_TOPOLOGY_TYPE, 1144
 TYPES::NODES_TYPE, 1152
 TYPES::REGION_TYPE, 1171
 NODES_CREATE_FINISH
 NODE_ROUTINES, 580
 NODES_CREATE_START
 NODE_ROUTINES, 580
 NODES_FINALISE
 NODE_ROUTINES, 581
 NODES_FINISHED
 TYPES::NODES_TYPE, 1152
 NODES_IN_FACE
 TYPES::DOMAIN_FACE_TYPE, 1002
 NODES_IN_LINE
 TYPES::DOMAIN_LINE_TYPE, 1008
 NODES_INITIALISE
 NODE_ROUTINES, 581

 NONLINEAR_DATA
 TYPES::EQUATIONS_TYPE, 1078
 NONLINEAR_SOLVER
 TYPES::SOLVER_TYPE, 1197
 MATHS
 MATHS::NORMALISE, 882
 MATHS::NORMALISE_DP
 MATHS, 490
 MATHS::NORMALISE_SP
 MATHS, 490
 MATHS::NORMALISE, 882
 NUM_BLOCKS
 COMP_ENVIRONMENT::MPI_-
 COMPUTATIONAL_NODE_TYPE,
 739
 NUM_BYTES
 BINARY_FILE::BINARY_TAG_TYPE, 702
 NUM_HEADER_BYTES
 BINARY_FILE::BINARY_TAG_TYPE, 702
 NUM_SUBTAGS
 BINARY_FILE::BINARY_TAG_TYPE, 702
 NUMBER
 TYPES::DECOMPOSITION_LINE_TYPE,
 965
 TYPES::DOMAIN_ELEMENT_TYPE, 997
 TYPES::DOMAIN_FACE_TYPE, 1003
 TYPES::DOMAIN_LINE_TYPE, 1008
 NUMBER_BASIS_FUNCTIONS
 TYPES::BASIS_FUNCTIONS_TYPE, 946
 NUMBER_COMPUTATIONAL_NODES
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 ENVIRONMENT_TYPE, 736
 NUMBER_IN_LIST
 LISTS::LIST_TYPE, 868
 NUMBER_IN_TREE
 TREES::TREE_TYPE, 944
 NUMBER_LEVELS
 COMP_ENVIRONMENT::CACHE_TYPE,
 734
 NUMBER_NON_ZEROS
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 978
 TYPES::MATRIX_TYPE, 1132
 NUMBER_OF_ADJACENT_DOMAINS
 TYPES::DOMAIN_MAPPING_TYPE, 1013
 NUMBER_OF_ADJACENT_ELEMENTS
 TYPES::DECOMPOSITION_ELEMENT_-
 TYPE, 962
 TYPES::MESH_ELEMENT_TYPE, 1135
 NUMBER_OF AREAS
 TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087
 NUMBER_OF_BOUNDARY
 TYPES::DOMAIN_MAPPING_TYPE, 1013
 NUMBER_OF_COLLAPSED_XI

TYPES::BASIS_TYPE, 955
NUMBER_OF_COLUMNS
 TYPES::ELEMENT_MATRIX_TYPE, 1028
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048
 TYPES::EQUATIONS_MATRIX_TYPE,
 1050
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1184
 TYPES::SOLVER_MATRIX_TYPE, 1192
NUMBER_OF_COMPONENTS
 FIELD_IO_ROUTINES::FIELD_IO_INFO_-
 SET, 763
 TYPES::FIELD_CREATE_VALUES_-
 CACHE_TYPE, 1080
 TYPES::FIELD_VARIABLE_TYPE, 1118
 TYPES::MESH_TYPE, 1147
NUMBER_OF_CONSTANT_DOFS
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
NUMBER_OF_CONSTANT_PARAMETERS
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097
NUMBER_OF_COORDINATE_SYSTEMS
 COORDINATE_-
 ROUTINES::COORDINATE_-
 SYSTEMS_TYPE, 753
NUMBER_OF_DECOMPOSITIONS
 TYPES::DECOMPOSITIONS_TYPE, 973
NUMBER_OF_DERIVATIVES
 TYPES::BASIS_TYPE, 955
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::MESH_NODE_TYPE, 1138
NUMBER_OF_DIMENSIONS
 TYPES::COORDINATE_SYSTEM_TYPE,
 959
 TYPES::DOMAIN_TYPE, 1026
 TYPES::MESH_TYPE, 1147
NUMBER_OF_DOFS
 TYPES::DOMAIN_DOFS_TYPE, 995
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
 TYPES::FIELD_VARIABLE_TYPE, 1118
 TYPES::MESH_DOFS_TYPE, 1133
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1184
NUMBER_OF_DOMAIN_LOCAL
 TYPES::DOMAIN_MAPPING_TYPE, 1014
NUMBER_OF_DOMAINS
 TYPES::DECOMPOSITION_TYPE, 972
 TYPES::DOMAIN_GLOBAL_MAPPING_-
 TYPE, 1006
 TYPES::DOMAIN_MAPPING_TYPE, 1014
NUMBER_OF_EDGES_CUT
 TYPES::DECOMPOSITION_TYPE, 972
 NUMBER_OF_ELEMENT_DOFS
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
 NUMBER_OF_ELEMENT_PARAMETERS
 TYPES::BASIS_TYPE, 955
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1097
 NUMBER_OF_ELEMENTS
 FIELD_IO_ROUTINES::FIELD_IO_-
 ELEMENTALL_INFO_SET, 762
 TYPES::DOMAIN_ELEMENTS_TYPE,
 1000
 TYPES::MESH_ELEMENTS_TYPE, 1137
 TYPES::MESH_TYPE, 1147
NUMBER_OF_ELEMENTS_XI
 TYPES::GENERATED_MESH_-
 REGULAR_TYPE, 1121
NUMBER_OF_EMBEDDED_MESHES
 TYPES::MESH_TYPE, 1147
NUMBER_OF_EQUATIONS_MATRICES
 TYPES::EQUATIONS_MAPPING_TYPE,
 1041
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 MAP_TYPE, 1181
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1203
NUMBER_OF_EQUATIONS_SET_-
 LINEARITY_TYPES
 EQUATIONS_SET_CONSTANTS_-
 LinearityTypes, 49
NUMBER_OF_EQUATIONS_SET_SOLUTION_-
 METHODS
 EQUATIONS_SET_CONSTANTS_-
 SolutionMethods, 54
NUMBER_OF_EQUATIONS_SET_TIME_-
 TYPES
 EQUATIONS_SET_CONSTANTS_-
 TimeDepedenceTypes, 51
NUMBER_OF_EQUATIONS_SETS
 TYPES::EQUATIONS_SETS_TYPE, 1068
 TYPES::SOLUTION_MAPPING_TYPE,
 1175
 TYPES::SOLVER_COL_TO_VARIABLE_-
 MAP_TYPE, 1187
NUMBER_OF_FACES
 TYPES::DOMAIN_FACES_TYPE, 1004
 TYPES::MESH_TYPE, 1147
NUMBER_OF_FIELDS
 TYPES::FIELDS_TYPE, 1119
NUMBER_OF_FIELDS_USING

TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087
 NUMBER_OF_GAUSS
 TYPES::QUADRATURE_SCHEME_TYPE,
 1166
 NUMBER_OF_GAUSS_XI
 TYPES::QUADRATURE_TYPE, 1168
 NUMBER_OF_GHOST
 TYPES::DOMAIN_MAPPING_TYPE, 1014
 NUMBER_OF_GLOBAL
 TYPES::DOMAIN_MAPPING_TYPE, 1014
 NUMBER_OF_INTERNAL
 TYPES::DOMAIN_MAPPING_TYPE, 1014
 NUMBER_OF_INVOCATIONS
 BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 692
 NUMBER_OF_ITERATIONS
 TYPES::PROBLEM_NONLINEAR_DATA_-
 TYPE, 1157
 NUMBER_OF_LINES
 TYPES::DECOMPOSITION_LINES_TYPE,
 967
 TYPES::DOMAIN_LINES_TYPE, 1010
 TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087
 TYPES::MESH_TYPE, 1147
 NUMBER_OF_LOCAL
 TYPES::DOMAIN_MAPPING_TYPE, 1014
 NUMBER_OF_LOCAL_LINES
 TYPES::BASIS_TYPE, 955
 NUMBER_OF_MATRICES
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
 TYPES::SOLVER_MATRICES_TYPE, 1189
 NUMBER_OF_MATRIX_VARIABLES
 TYPES::EQUATIONS_MAPPING_TYPE,
 1041
 NUMBER_OF_MESHES
 TYPES::MESHES_TYPE, 1149
 NUMBER_OF_NODE_DOFS
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
 NUMBER_OF_NODE_LINES
 TYPES::DOMAIN_NODE_TYPE, 1018
 NUMBER_OF_NODE_PARAMETERS
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1098
 NUMBER_OF_NODES
 FIELD_IO_ROUTINES::FIELD_IO_-
 NODAL_INFO_SET, 764
 TYPES::BASIS_TYPE, 955
 TYPES::DOMAIN_NODES_TYPE, 1021
 TYPES::MESH_NODES_TYPE, 1140
 TYPES::NODES_TYPE, 1153

 NUMBER_OF_NODES_IN_LOCAL_LINE
 TYPES::BASIS_TYPE, 955
 NUMBER_OF_NODES_XI
 TYPES::BASIS_TYPE, 955
 NUMBER_OF_PARAMETER_SETS
 TYPES::FIELD_PARAMETER_SETS_-
 TYPE, 1102
 NUMBER_OF_PARAMETERS
 TYPES::FIELD_INTERPOLATION_-
 PARAMETERS_TYPE, 1094
 NUMBER_OF_PARTIAL_DERIVATIVES
 TYPES::BASIS_TYPE, 956
 NUMBER_OF_POINT_DOFS
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
 NUMBER_OF_POINT_PARAMETERS
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1098
 NUMBER_OF_PROBLEM_LINEARITIES
 PROBLEM_ROUTINES_LinearityTypes, 108
 NUMBER_OF_PROBLEM_TIME_TYPES
 PROBLEM_ROUTINES_-
 TimeDepedenceTypes, 110
 NUMBER_OF_PROBLEMS
 TYPES::PROBLEMS_TYPE, 1163
 NUMBER_OF_RECEIVE_GHOSTS
 TYPES::DOMAIN_ADJACENT_-
 DOMAIN_TYPE, 994
 NUMBER_OF_ROWS
 TYPES::ELEMENT_MATRIX_TYPE, 1029
 TYPES::ELEMENT_VECTOR_TYPE, 1030
 TYPES::EQUATIONS_MAPPING_TYPE,
 1041
 TYPES::EQUATIONS_MATRICES_TYPE,
 1044
 TYPES::SOLUTION_MAPPING_TYPE,
 1175
 TYPES::SOLVER_MATRICES_TYPE, 1189
 TYPES::SOLVER_ROW_TO_-
 EQUATIONS_SET_MAP_TYPE,
 1194
 NUMBER_OF_SCALING_INDICES
 TYPES::FIELD_SCALINGS_TYPE, 1107
 NUMBER_OF_SCHEMES
 TYPES::QUADRATURE_TYPE, 1168
 NUMBER_OF_SEND_GHOSTS
 TYPES::DOMAIN_ADJACENT_-
 DOMAIN_TYPE, 994
 NUMBER_OF_SOLUTIONS
 TYPES::PROBLEM_TYPE, 1161
 NUMBER_OF_SOLVER_COLS
 TYPES::EQUATIONS_COL_TO_SOLVER_-
 COLS_MAP_TYPE, 1031
 NUMBER_OF_SOLVER_MATRICES

TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_EM_TYPE, 1073
 TYPES::SOLUTION_MAPPING_TYPE,
 1175
 NUMBER_OF_SOLVER_ROWS
 TYPES::EQUATIONS_ROW_TO_-
 SOLVER_ROWS_MAP_TYPE, 1052
 NUMBER_OF_SUB_BASES
 TYPES::BASIS_TYPE, 956
 NUMBER_OF_SUB_REGIONS
 TYPES::REGION_TYPE, 1171
 NUMBER_OF_SURROUNDING_ELEMENTS
 TYPES::DECOMPOSITION_LINE_TYPE,
 966
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::MESH_NODE_TYPE, 1139
 NUMBER_OF_VARIABLES
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076
 TYPES::FIELD_TYPE, 1110
 NUMBER_OF_VOLUMES
 TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087
 NUMBER_OF_X_DIMENSIONS
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
 NUMBER_OF_XI
 TYPES::BASIS_TYPE, 956
 NUMBER_OF_XI_COORDINATES
 TYPES::BASIS_TYPE, 956
 NUMBER_OF_XI_DIMENSIONS
 TYPES::FIELD_INTERPOLATED_POINT_-
 METRICS_TYPE, 1089
 NUMBER_PROCESSORS
 COMP_ENVIRONMENT::COMPUTATIONAL_op_eq_CH_VS
 NODE_TYPE, 737
 NUMBER_TO_CHARACTER_DP
 STRINGS, 656
 STRINGS::NUMBER_TO_CHARACTER,
 930
 NUMBER_TO_CHARACTER_INTG
 STRINGS, 656
 STRINGS::NUMBER_TO_CHARACTER,
 930
 NUMBER_TO_CHARACTER_LINTG
 STRINGS, 657
 STRINGS::NUMBER_TO_CHARACTER,
 930
 NUMBER_TO_CHARACTER_SP
 STRINGS, 657
 STRINGS::NUMBER_TO_CHARACTER,
 931
 NUMBER_TO_VSTRING_DP
 STRINGS, 658
 STRINGS::NUMBER_TO_VSTRING, 932
 NUMBER_TO_VSTRING_INTG
 STRINGS, 658
 STRINGS::NUMBER_TO_VSTRING, 932
 NUMBER_TO_VSTRING_LINTG
 STRINGS, 659
 STRINGS::NUMBER_TO_VSTRING, 932
 NUMBER_TO_VSTRING_SP
 STRINGS, 659
 STRINGS::NUMBER_TO_VSTRING, 933
 OFFDIAGONAL_NUMBER_NON_ZEROS
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 978
 op_assign_CH_VS
 ISO_VARYING_STRING, 443
 ISO_VARYING_STRING::assignment(=),
 817
 op_assign_VS_CH
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 817
 op_concat_CH_VS
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 817
 op_concat_VS_CH
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 818
 op_concat_VS_VS
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 818
 op_eq_CH_VS
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 818
 op_eq_VS_CH
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 818
 op_eq_VS_VS
 ISO_VARYING_STRING, 444
 ISO_VARYING_STRING::assignment(=),
 818
 op_ge_CH_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 818
 op_ge_VS_CH
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 818

op_ge_VS_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 818

op_gt_CH_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 818

op_gt_VS_CH
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 819

op_gt_VS_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 819

op_le_CH_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 819

op_le_VS_CH
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 819

op_le_VS_VS
 ISO_VARYING_STRING, 445
 ISO_VARYING_STRING::assignment(=),
 819

op_lt_CH_VS
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 819

op_lt_VS_CH
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 819

op_lt_VS_VS
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 819

op_ne_CH_VS
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 819

op_ne_VS_CH
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 820

op_ne_VS_VS
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::assignment(=),
 820

OP_STRING
 BASE_ROUTINES, 173

OPEN_BINARY_FILE
 BINARY_FILE, 178

OPEN_CMISS_BINARY_FILE
 BINARY_FILE, 178

OPEN_COMFILE_UNIT
 BASE_ROUTINES_FileUnits, 24

ORIENTATION
 TYPES::COORDINATE_SYSTEM_TYPE,
 959

ORIGIN
 TYPES::COORDINATE_SYSTEM_TYPE,
 959

TYPES::GENERATED_MESH_-
 REGULAR_TYPE, 1121

OS_TYPE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 699

OUTPUT_SET_OFF
 BASE_ROUTINES, 168

OUTPUT_SET_ON
 BASE_ROUTINES, 168

OUTPUT_TYPE
 TYPES::EQUATIONS_TYPE, 1078
 TYPES::SOLVER_TYPE, 1197

PACKCHARACTERS
 F90C::interface, 760

PackCharacters
 f90c_c.c, 1272

PARAM_TO_DOF_MAP
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1113

PARAMETER_SETS
 TYPES::FIELD_PARAMETER_SETS_-
 TYPE, 1102

TYPES::FIELD_TYPE, 1110

PARAMETERS
 TYPES::FIELD_INTERPOLATION_-
 PARAMETERS_TYPE, 1094

TYPES::FIELD_PARAMETER_SET_TYPE,
 1100

PARENT
 TREES::TREE_NODE_TYPE, 942

PARENT_BASIS
 TYPES::BASIS_TYPE, 956

PARENT_REGION
 TYPES::REGION_TYPE, 1171

PARMETIS_PARTKWAY
 CMISS_PARMETIS, 198

PARMETIS_PARTMESHKWAY
 CMISS_PARMETIS, 198

ParMETIS_V3_PartK
 CMISS_PARMETIS::interface, 716

ParMETIS_V3_PartMeshKway

CMISS_PARMETIS::interface, 716
PARTIAL_DERIVATIVE_INDEX
 TYPES::BASIS_TYPE, 956
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::MESH_NODE_TYPE, 1139
PARTIAL_DERIVATIVE_TYPE
 TYPES::FIELD_INTERPOLATED_POINT_TYPE, 1091
PC
 TYPES::LINEAR_ITERATIVE_SOLVER_TYPE, 1125
PCSetType
 CMISS_PETSC::interface, 722
PETSC
 TYPES::DISTRIBUTED_MATRIX_TYPE, 980
 TYPES::DISTRIBUTED_VECTOR_TYPE, 992
PETSC_ERRORHANDLING_SET_OFF
 CMISS_PETSC, 205
PETSC_ERRORHANDLING_SET_ON
 CMISS_PETSC, 205
PETSC_FINALIZE
 CMISS_PETSC, 206
PETSC_HANDLE_ERROR
 CMISS_PETSC, 239
PETSC_INITIALIZE
 CMISS_PETSC, 206
PETSC_ISDESTROY
 CMISS_PETSC, 206
PETSC_ISFINALISE
 CMISS_PETSC, 207
PETSC_ISINITIALISE
 CMISS_PETSC, 207
PETSC_ISLOCALTOGLOBALMAPPINGAPPLY
 CMISS_PETSC, 207
PETSC_ISLOCALTOGLOBALMAPPINGAPPLYIS
 CMISS_PETSC, 208
PETSC_ISLOCALTOGLOBALMAPPINGCREATE
 CMISS_PETSC, 208
PETSC_ISLOCALTOGLOBALMAPPINGDESTROYPETSC_MATGETARRAY
 CMISS_PETSC, 209
PETSC_ISLOCALTOGLOBALMAPPINGFINALISEPETSC_MATGETOWNERSHIPRANGE
 CMISS_PETSC, 209
PETSC_ISLOCALTOGLOBALMAPPINGINITIALISEPETSC_MATGETVALUES
 CMISS_PETSC, 209
PETSC_KSPCREATE
 CMISS_PETSC, 210
PETSC_KSPDESTROY
 CMISS_PETSC, 210
PETSC_KSPFINALISE
 CMISS_PETSC, 211
PETSC_KSPGETCONVERGEDREASON
 CMISS_PETSC, 211
PETSC_KSPGETITERATIONNUMBER
 CMISS_PETSC, 211
PETSC_KSPGETPC
 CMISS_PETSC, 212
PETSC_KSPGETRESIDUALNORM
 CMISS_PETSC, 212
PETSC_KSPINITIALISE
 CMISS_PETSC, 213
PETSC_KSPSETFROMOPTIONS
 CMISS_PETSC, 213
PETSC_KSPSETOPERATORS
 CMISS_PETSC, 213
PETSC_KSPSETTOLERANCES
 CMISS_PETSC, 214
PETSC_KSPSETTYPE
 CMISS_PETSC, 214
PETSC_KSPSETUP
 CMISS_PETSC, 215
PETSC_KSPSOLVE
 CMISS_PETSC, 215
PETSC_LOGPRINTSUMMARY
 CMISS_PETSC, 216
PETSC_MATASEMBLYBEGIN
 CMISS_PETSC, 216
PETSC_MATASEMBLYEND
 CMISS_PETSC, 216
PETSC_MATCREATE
 CMISS_PETSC, 217
PETSC_MATCREATEMPIAIJ
 CMISS_PETSC, 217
PETSC_MATCREATEMPIDENSE
 CMISS_PETSC, 218
PETSC_MATCREATESEQAIJ
 CMISS_PETSC, 218
PETSC_MATCREATESEQDENSE
 CMISS_PETSC, 219
PETSC_MATDESTROY
 CMISS_PETSC, 219
PETSC_MATFINALISE
 CMISS_PETSC, 220
PETSC_MATGETARRAY
 CMISS_PETSC, 220
PETSC_MATINITIALISE
 CMISS_PETSC, 221
PETSC_MATRESTOREARRAY
 CMISS_PETSC, 221
PETSC_MATSETLOCALTOGLOBALMAPPING
 CMISS_PETSC, 222
PETSC_MATSETOPTION
 CMISS_PETSC, 222

PETSC_MATSETSIZES
 CMISS_PETSC, 223
 PETSC_MATSETVALUE
 CMISS_PETSC, 223
 PETSC_MATSETVALUELOCAL
 CMISS_PETSC, 223
 PETSC_MATSETVALUES
 CMISS_PETSC, 224
 PETSC_MATSETVALUESLOCAL
 CMISS_PETSC, 224
 PETSC_MATVIEW
 CMISS_PETSC, 225
 PETSC_MATZEROENTRIES
 CMISS_PETSC, 225
 PETSC_PCFINALISE
 CMISS_PETSC, 226
 PETSC_PCINITIALISE
 CMISS_PETSC, 226
 PETSC_PCSETTYPE
 CMISS_PETSC, 226
 PETSC_VECASSEMBLYBEGIN
 CMISS_PETSC, 227
 PETSC_VECASSEMBLYEND
 CMISS_PETSC, 227
 PETSC_VECCREATE
 CMISS_PETSC, 228
 PETSC_VECCREATEGHOST
 CMISS_PETSC, 228
 PETSC_VECCREATEGHOSTWITHARRAY
 CMISS_PETSC, 228
 PETSC_VECCREATEMPI
 CMISS_PETSC, 229
 PETSC_VECCREATEMPIWITHARRAY
 CMISS_PETSC, 229
 PETSC_VECCREATESEQ
 CMISS_PETSC, 230
 PETSC_VECCREATESEQWITHARRAY
 CMISS_PETSC, 230
 PETSC_VECDESTROY
 CMISS_PETSC, 231
 PETSC_VECDUPLICATE
 CMISS_PETSC, 231
 PETSC_VECFINALISE
 CMISS_PETSC, 231
 PETSC_VECGETARRAY
 CMISS_PETSC, 232
 PETSC_VECGETARRAYF90
 CMISS_PETSC, 232
 PETSC_VECGETLOCALSIZE
 CMISS_PETSC, 232
 PETSC_VECGETOWNERSHIPRANGE
 CMISS_PETSC, 233
 PETSC_VECGETSIZE
 CMISS_PETSC, 233
 PETSC_VECGETVALUES
 CMISS_PETSC, 234
 PETSC_VECGHOSTGETLOCALFORM
 CMISS_PETSC, 234
 PETSC_VECGHOSTRESTORELOCALFORM
 CMISS_PETSC, 234
 PETSC_VECGHOSTUPDATEBEGIN
 CMISS_PETSC, 235
 PETSC_VECGHOSTUPDATEEND
 CMISS_PETSC, 235
 PETSC_VECINITIALISE
 CMISS_PETSC, 236
 PETSC_VECRESTOREARRAY
 CMISS_PETSC, 236
 PETSC_VECRESTOREARRAYF90
 CMISS_PETSC, 236
 PETSC_VECSET
 CMISS_PETSC, 237
 PETSC_VECSETFROMOPTIONS
 CMISS_PETSC, 237
 PETSC_VECSETLOCALTOGLOBALMAPPING
 CMISS_PETSC, 237
 PETSC_VECSETSIZES
 CMISS_PETSC, 238
 PETSC_VECSETVALUES
 CMISS_PETSC, 238
 PETSC_VECSETVALUESLOCAL
 CMISS_PETSC, 239
 PETSC_VECVIEW
 CMISS_PETSC, 239
 PetscFinalize
 CMISS_PETSC::interface, 722
 PetscInitialize
 CMISS_PETSC::interface, 723
 PetscLogPrintSummary
 CMISS_PETSC::interface, 723
 POINT_DOF2PARAM_MAP
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1084
 POINT_PARAM2DOF_MAP
 TYPES::FIELD_PARAM_TO_DOF_MAP_-
 TYPE, 1098
 PREVIOUS_ROUTINE
 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695
 PROBLEM
 TYPES::PROBLEM_CONTROL_TYPE,
 1155
 TYPES::SOLUTION_TYPE, 1178
 PROBLEM_ADVECTION_DIFFUSION_-
 EQUATION_TYPE
 PROBLEM_CONSTANTS, 583
 PROBLEM_BIHAMONIC_EQUATION_TYPE
 PROBLEM_CONSTANTS, 583

PROBLEM_CLASSICAL_FIELD_CLASS
 PROBLEM_CONSTANTS, 584

PROBLEM_CONSTANTS, 582
 PROBLEM_ADVECTION_DIFFUSION_-
 EQUATION_TYPE, 583

 PROBLEM_BIHAMONIC_EQUATION_-
 TYPE, 583

 PROBLEM_CLASSICAL_FIELD_CLASS,
 584

 PROBLEM_DIFFUSION_EQUATION_-
 TYPE, 584

 PROBLEM_ELASTICITY_CLASS, 584

 PROBLEM_ELECTROMAGNETICS_-
 CLASS, 584

 PROBLEM_ELECTROSTATIC_TYPE, 584

 PROBLEMFINITE_ELASTICITY_TYPE,
 584

 PROBLEM_FITTING_CLASS, 584

 PROBLEM_FLUID_MECHANICS_CLASS,
 584

 PROBLEM_GENERALISED_LAPLACE_-
 SUBTYPE, 585

 PROBLEM_HELMHOLTZ_EQUATION_-
 TYPE, 585

 PROBLEM_LAPLACE_EQUATION_TYPE,
 585

 PROBLEM_LINEAR_ELASTIC_MODAL_-
 TYPE, 585

 PROBLEM_LINEAR_ELASTICITY_TYPE,
 585

 PROBLEM_MAGNETOSTATIC_TYPE, 585

 PROBLEM_MAXWELLS_EQUATIONS_-
 TYPE, 585

 PROBLEM_MODAL_CLASS, 586

 PROBLEM_NAVIER_STOKES_FLUID_-
 TYPE, 586

 PROBLEM_NO_CLASS, 586

 PROBLEM_NO_SUBTYPE, 586

 PROBLEM_NO_TYPE, 586

 PROBLEM_OPTIMISATION_CLASS, 586

 PROBLEM_POISSON_EQUATION_TYPE,
 586

 PROBLEM_REACTION_DIFFUSION_-
 EQUATION_TYPE, 586

 PROBLEM_STANDARD_LAPLACE_-
 SUBTYPE, 587

 PROBLEM_STOKES_FLUID_TYPE, 587

 PROBLEM_WAVE_EQUATION_TYPE, 587

PROBLEM_CONSTANTS::SetupActionTypes,
 101

PROBLEM_CONSTANTS::SetupTypes, 99

PROBLEM_CONSTANTS::SolutionOutputTypes,
 103

PROBLEM_CONSTANTS::SolverOutputTypes,
 105

PROBLEM_CONSTANTS::SolverSparsityTypes,
 107

PROBLEM_CONSTANTS_SetupActionTypes
 PROBLEM_SETUP_DO_ACTION, 101

 PROBLEM_SETUP_FINISH_ACTION, 101

 PROBLEM_SETUP_START_ACTION, 102

PROBLEM_CONSTANTS_SetupTypes
 PROBLEM_SETUP_CONTROL_TYPE, 99

 PROBLEM_SETUP_INITIAL_TYPE, 99

 PROBLEM_SETUP SOLUTION_TYPE, 100

 PROBLEM_SETUP_SOLVER_TYPE, 100

PROBLEM_CONSTANTS_SolutionOutputTypes
 PROBLEM SOLUTION MATRIX_-
 OUTPUT, 103

 PROBLEM SOLUTION_NO_OUTPUT, 103

 PROBLEM SOLUTION_TIMING_-
 OUTPUT, 103

PROBLEM_CONSTANTS_SolverOutputTypes
 PROBLEM SOLVER_NO_OUTPUT, 105

 PROBLEM SOLVER_SOLVER_OUTPUT,
 105

 PROBLEM SOLVER_TIMING_OUTPUT,
 105

PROBLEM_CONSTANTS_SolverSparsityTypes
 PROBLEM SOLVER_FULL_MATRICES,
 107

 PROBLEM SOLVER_SPARSE_-
 MATRICES, 107

PROBLEM_CONTROL_CREATE_FINISH
 PROBLEM_ROUTINES, 590

PROBLEM_CONTROL_CREATE_START
 PROBLEM_ROUTINES, 591

PROBLEM_CONTROL_DESTROY
 PROBLEM_ROUTINES, 591

PROBLEM_CONTROL_FINALISE
 PROBLEM_ROUTINES, 591

PROBLEM_CONTROL_INITIALISE
 PROBLEM_ROUTINES, 592

PROBLEM_CREATE_FINISH
 PROBLEM_ROUTINES, 592

PROBLEM_CREATE_START
 PROBLEM_ROUTINES, 593

PROBLEM_DESTROY_NUMBER
 PROBLEM_ROUTINES, 593

 PROBLEM_ROUTINES::PROBLEM_-
 DESTROY, 915

PROBLEM_DESTROY_PTR
 PROBLEM_ROUTINES, 594

 PROBLEM_ROUTINES::PROBLEM_-
 DESTROY, 915

PROBLEM_DIFFUSION_EQUATION_TYPE
 PROBLEM_CONSTANTS, 584

PROBLEM_DYNAMIC
 PROBLEM_ROUTINES_-
 TimeDepedenceTypes, 110
PROBLEM_ELASTICITY_CLASS
 PROBLEM_CONSTANTS, 584
PROBLEM_ELECTROMAGNETICS_CLASS
 PROBLEM_CONSTANTS, 584
PROBLEM_ELECTROSTATIC_TYPE
 PROBLEM_CONSTANTS, 584
PROBLEM_FINALISE
 PROBLEM_ROUTINES, 594
PROBLEM_FINISHED
 TYPES::PROBLEM_TYPE, 1161
PROBLEMFINITE_ELASTICITY_TYPE
 PROBLEM_CONSTANTS, 584
PROBLEM_FITTING_CLASS
 PROBLEM_CONSTANTS, 584
PROBLEM_FLUID_MECHANICS_CLASS
 PROBLEM_CONSTANTS, 584
PROBLEM_GENERALISED_LAPLACE_-SUBTYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_HELMHOLTZ_EQUATION_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_INITIALISE
 PROBLEM_ROUTINES, 594
PROBLEM_LAPLACE_EQUATION_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_LINEAR
 PROBLEM_ROUTINES_LinearityTypes, 108
PROBLEM_LINEAR_ELASTIC_MODAL_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_LINEAR_ELASTICITY_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_MAGNETOSTATIC_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_MAXWELLS_EQUATIONS_TYPE
 PROBLEM_CONSTANTS, 585
PROBLEM_MODAL_CLASS
 PROBLEM_CONSTANTS, 586
PROBLEM_NAVIER_STOKES_FLUID_TYPE
 PROBLEM_CONSTANTS, 586
PROBLEM_NO_CLASS
 PROBLEM_CONSTANTS, 586
PROBLEM_NO_SUBTYPE
 PROBLEM_CONSTANTS, 586
PROBLEM_NO_TYPE
 PROBLEM_CONSTANTS, 586
PROBLEM_NONLINEAR
 PROBLEM_ROUTINES_LinearityTypes, 108
PROBLEM_NONLINEAR_BCS
 PROBLEM_ROUTINES_LinearityTypes, 109
PROBLEM_OPTIMISATION_CLASS
 PROBLEM_CONSTANTS, 586

PROBLEM_POISSON_EQUATION_TYPE
 PROBLEM_CONSTANTS, 586
PROBLEM_QUASISTATIC
 PROBLEM_ROUTINES_-
 TimeDepedenceTypes, 110
PROBLEM_REACTION_DIFFUSION_EQUATION_TYPE
 PROBLEM_CONSTANTS, 586
PROBLEM_ROUTINES, 588
 PROBLEM_CONTROL_CREATE_FINISH,
 590
 PROBLEM_CONTROL_CREATE_START,
 591
 PROBLEM_CONTROL_DESTROY, 591
 PROBLEM_CONTROL_FINALISE, 591
 PROBLEM_CONTROL_INITIALISE, 592
 PROBLEM_CREATE_FINISH, 592
 PROBLEM_CREATE_START, 593
 PROBLEM_DESTROY_NUMBER, 593
 PROBLEM_DESTROY_PTR, 594
 PROBLEM_FINALISE, 594
 PROBLEM_INITIALISE, 594
 PROBLEM_SETUP, 595
PROBLEM_SOLUTION_EQUATIONS_SET_ADD, 595
 PROBLEM_SOLUTION_FINALISE, 596
 PROBLEM_SOLUTION_INITIALISE, 596
 PROBLEM_SOLUTION_SOLVE, 597
 PROBLEM_SOLUTIONS_CREATE_FINISH, 597
 PROBLEM_SOLUTIONS_CREATE_START, 598
 PROBLEM_SOLUTIONS_FINALISE, 598
 PROBLEM_SOLUTIONS_INITIALISE, 598
 PROBLEM_SOLVE, 599
 PROBLEM_SOLVER_CREATE_FINISH,
 599
 PROBLEM_SOLVER_CREATE_START, 600
 PROBLEM_SOLVER_DESTROY, 600
 PROBLEM_SOLVER_GET, 600
 PROBLEM_SPECIFICATION_SET_NUMBER, 601
 PROBLEM_SPECIFICATION_SET_PTR,
 601
 PROBLEM_USER_NUMBER_FIND, 602
PROBLEMS, 603
 PROBLEMS_FINALISE, 602
 PROBLEMS_INITIALISE, 603
PROBLEM_ROUTINES::LinearityTypes, 108
PROBLEM_ROUTINES::PROBLEM_DESTROY,
 915
 PROBLEM_DESTROY_NUMBER, 915
 PROBLEM_DESTROY_PTR, 915

PROBLEM_ROUTINES::PROBLEM_-
SPECIFICATION_SET, 916
PROBLEM_SPECIFICATION_SET_-
NUMBER, 916
PROBLEM_SPECIFICATION_SET_PTR,
916
PROBLEM_ROUTINES::TimeDepedenceTypes,
110
PROBLEM_ROUTINES_LinearityTypes
NUMBER_OF_PROBLEM_LINEARITIES,
108
PROBLEM_LINEAR, 108
PROBLEM_NONLINEAR, 108
PROBLEM_NONLINEAR_BCS, 109
PROBLEM_ROUTINES_TimeDepedenceTypes
NUMBER_OF_PROBLEM_TIME_TYPES,
110
PROBLEM_DYNAMIC, 110
PROBLEM_QUASISTATIC, 110
PROBLEM_STATIC, 111
PROBLEM_SETUP
PROBLEM_ROUTINES, 595
PROBLEM_SETUP_CONTROL_TYPE
PROBLEM_CONSTANTS_SetupTypes, 99
PROBLEM_SETUP_DO_ACTION
PROBLEM_CONSTANTS_-
SetupActionTypes, 101
PROBLEM_SETUP_FINISH_ACTION
PROBLEM_CONSTANTS_-
SetupActionTypes, 101
PROBLEM_SETUP_INITIAL_TYPE
PROBLEM_CONSTANTS_SetupTypes, 99
PROBLEM_SETUP SOLUTION_TYPE
PROBLEM_CONSTANTS_SetupTypes, 100
PROBLEM_SETUP_SOLVER_TYPE
PROBLEM_CONSTANTS_SetupTypes, 100
PROBLEM_SETUP_START_ACTION
PROBLEM_CONSTANTS_-
SetupActionTypes, 102
PROBLEM_SOLUTION_EQUATIONS_SET_-
ADD
PROBLEM_ROUTINES, 595
PROBLEM_SOLUTION_FINALISE
PROBLEM_ROUTINES, 596
PROBLEM_SOLUTION_INITIALISE
PROBLEM_ROUTINES, 596
PROBLEM_SOLUTION_MATRIX_OUTPUT
PROBLEM_CONSTANTS_-
SolutionOutputTypes, 103
PROBLEM_SOLUTION_NO_OUTPUT
PROBLEM_CONSTANTS_-
SolutionOutputTypes, 103
PROBLEM_SOLUTION_solve
PROBLEM_ROUTINES, 597
PROBLEM_SOLUTION_TIMING_OUTPUT
PROBLEM_CONSTANTS_-
SolutionOutputTypes, 103
PROBLEM_SOLUTIONS_CREATE_FINISH
PROBLEM_ROUTINES, 597
PROBLEM_SOLUTIONS_CREATE_START
PROBLEM_ROUTINES, 598
PROBLEM_SOLUTIONS_FINALISE
PROBLEM_ROUTINES, 598
PROBLEM_SOLUTIONS_INITIALISE
PROBLEM_ROUTINES, 598
PROBLEM_SOLVE
PROBLEM_ROUTINES, 599
PROBLEM_SOLVER_CREATE_FINISH
PROBLEM_ROUTINES, 599
PROBLEM_SOLVER_CREATE_START
PROBLEM_ROUTINES, 600
PROBLEM_SOLVER_DESTROY
PROBLEM_ROUTINES, 600
PROBLEM_SOLVER_FULL_MATRICES
PROBLEM_CONSTANTS_-
SolverSparsityTypes, 107
PROBLEM_SOLVER_GET
PROBLEM_ROUTINES, 600
PROBLEM_SOLVER_NO_OUTPUT
PROBLEM_CONSTANTS_-
SolverOutputTypes, 105
PROBLEM_SOLVER_SOLVER_OUTPUT
PROBLEM_CONSTANTS_-
SolverOutputTypes, 105
PROBLEM_SOLVER_SPARSE_MATRICES
PROBLEM_CONSTANTS_-
SolverSparsityTypes, 107
PROBLEM_SOLVER_TIMING_OUTPUT
PROBLEM_CONSTANTS_-
SolverOutputTypes, 105
PROBLEM_SPECIFICATION_SET_NUMBER
PROBLEM_ROUTINES, 601
PROBLEM_ROUTINES::PROBLEM_-
SPECIFICATION_SET, 916
PROBLEM_SPECIFICATION_SET_PTR
PROBLEM_ROUTINES, 601
PROBLEM_ROUTINES::PROBLEM_-
SPECIFICATION_SET, 916
PROBLEM_STANDARD_LAPLACE_SUBTYPE
PROBLEM_CONSTANTS, 587
PROBLEM_STATIC
PROBLEM_ROUTINES_-
TimeDepedenceTypes, 111
PROBLEM_STOKES_FLUID_TYPE
PROBLEM_CONSTANTS, 587
PROBLEM_USER_NUMBER_FIND
PROBLEM_ROUTINES, 602
PROBLEM_WAVE_EQUATION_TYPE

PROBLEM_CONSTANTS, 587
 PROBLEMS
 PROBLEM_ROUTINES, 603
 TYPES::PROBLEM_TYPE, 1161
 TYPES::PROBLEMS_TYPE, 1163
 PROBLEMS_FINALISE
 PROBLEM_ROUTINES, 602
 PROBLEMS_INITIALISE
 PROBLEM_ROUTINES, 603
 PTR
 COORDINATE_-
 ROUTINES::COORDINATE_-
 SYSTEM_PTR_TYPE, 750
 FIELD_IO_ROUTINES::FIELD_-
 VARIABLE_COMPONENT_PTR_-
 TYPE, 766
 FIELD_IO_ROUTINES::MESH_-
 ELEMENTS_TYPE_PTR_TYPE,
 767
 KINDS_IntegerKinds, 77
 LISTS::LIST_PTR_TYPE, 855
 TYPES::BASIS_PTR_TYPE, 947
 TYPES::DECOMPOSITION_PTR_TYPE,
 968
 TYPES::DOMAIN_FACE_PTR_TYPE, 1001
 TYPES::DOMAIN_LINE_PTR_TYPE, 1007
 TYPES::DOMAIN_PTR_TYPE, 1022
 TYPES::EQUATIONS_MATRIX_PTR_-
 TYPE, 1046
 TYPES::EQUATIONS_SET_PTR_TYPE,
 1060
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_PTR_TYPE, 1070
 TYPES::FIELD_PARAMETER_SET_PTR_-
 TYPE, 1099
 TYPES::FIELD_PTR_TYPE, 1104
 TYPES::FIELD_VARIABLE_PTR_TYPE,
 1115
 TYPES::MESH_PTR_TYPE, 1141
 TYPES::MESH_TOPOLOGY_PTR_TYPE,
 1142
 TYPES::PROBLEM_PTR_TYPE, 1158
 TYPES::QUADRATURE_SCHEME_PTR_-
 TYPE, 1164
 TYPES::REGION_PTR_TYPE, 1169
 TYPES::SOLUTION_PTR_TYPE, 1177
 TYPES::SOLVER_MATRIX_PTR_TYPE,
 1191
 put_CH
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::put, 835
 put_line_CH
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::put_line, 836
 put_line_unit_CH
 ISO_VARYING_STRING, 446
 ISO_VARYING_STRING::put_line, 836
 put_line_unit_VS
 ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::put_line, 836
 put_line_VS
 ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::put_line, 836
 put_unit_CH
 ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::put, 835
 put_unit_VS
 ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::put, 835
 put_VS
 ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::put, 835
 QP
 KINDS_RealKinds, 80
 QUADRATURE
 TYPES::BASIS_TYPE, 956
 TYPES::QUADRATURE_SCHEME_TYPE,
 1166
 QUADRATURE_SCHEME_MAP
 TYPES::QUADRATURE_TYPE, 1168
 QUADRUPLECOMPLEXTYPE
 binary_file_c.c, 1219
 QUADRUPLETYP
 binary_file_c.c, 1219
 RADIAL_INTERPOLATION_TYPE
 TYPES::COORDINATE_SYSTEM_TYPE,
 959
 RANK
 COMP_ENVIRONMENT::COMPUTATIONAL_-
 NODE_TYPE, 737
 READ_BINARY_FILE_CHARACTER
 BINARY_FILE, 179
 BINARY_FILE::READ_BINARY_FILE, 705
 READ_BINARY_FILE_DP
 BINARY_FILE, 179
 BINARY_FILE::READ_BINARY_FILE, 705
 READ_BINARY_FILE_DPI
 BINARY_FILE, 179
 BINARY_FILE::READ_BINARY_FILE, 705
 READ_BINARY_FILE_DPC
 BINARY_FILE, 179
 BINARY_FILE::READ_BINARY_FILE, 705
 READ_BINARY_FILE_DPC1
 BINARY_FILE, 180
 BINARY_FILE::READ_BINARY_FILE, 706
 READ_BINARY_FILE_INTG

BINARY_FILE, 180
BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_INTG1
 BINARY_FILE, 180
 BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_LINTG
 BINARY_FILE, 180
 BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_LINTG1
 BINARY_FILE, 180
 BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_LOGICAL
 BINARY_FILE, 181
 BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_LOGICAL1
 BINARY_FILE, 181
 BINARY_FILE::READ_BINARY_FILE, 706
READ_BINARY_FILE_SINTG
 BINARY_FILE, 181
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_FILE_SINTG1
 BINARY_FILE, 181
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_FILE_SP
 BINARY_FILE, 182
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_FILE_SP1
 BINARY_FILE, 182
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_FILE_SPC
 BINARY_FILE, 182
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_FILE_SPC1
 BINARY_FILE, 182
 BINARY_FILE::READ_BINARY_FILE, 707
READ_BINARY_TAG_HEADER
 BINARY_FILE, 182
RECEIVE_BUFFER_DP
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
RECEIVE_BUFFER_INTG
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
RECEIVE_BUFFER_L
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
RECEIVE_BUFFER_SIZE
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
RECEIVE_BUFFER_SP
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
RECEIVE_TAG_NUMBER
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
REGION
 TYPES::DOMAIN_TYPE, 1026
 TYPES::EQUATIONS_SET_TYPE, 1066
 TYPES::EQUATIONS_SETS_TYPE, 1068
 TYPES::FIELD_TYPE, 1110
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1114
 TYPES::FIELD_VARIABLE_TYPE, 1118
 TYPES::FIELDS_TYPE, 1119
 TYPES::GENERATED_MESH_TYPE, 1122
 TYPES::MESH_TYPE, 1147
 TYPES::MESHS_TYPE, 1149
 TYPES::NODES_TYPE, 1153
REGION_COORDINATE_SYSTEM_GET
 REGION_ROUTINES, 604
REGION_COORDINATE_SYSTEM_SET_-
 NUMBER
 REGION_ROUTINES, 605
 REGION_ROUTINES::REGION_-
 COORDINATE_SYSTEM_SET, 918
REGION_COORDINATE_SYSTEM_SET_PTR
 REGION_ROUTINES, 605
 REGION_ROUTINES::REGION_-
 COORDINATE_SYSTEM_SET, 918
REGION_CREATE_FINISH
 REGION_ROUTINES, 605
REGION_CREATE_START
 REGION_ROUTINES, 605
REGION_DESTROY
 REGION_ROUTINES, 606
REGION_FINISHED
 TYPES::REGION_TYPE, 1171
REGION_LABEL_GET
 REGION_ROUTINES, 606
REGION_LABEL_SET_NUMBER
 REGION_ROUTINES, 606
 REGION_ROUTINES::REGION_LABEL_-
 SET, 919
REGION_LABEL_SET_PTR
 REGION_ROUTINES, 606
 REGION_ROUTINES::REGION_LABEL_-
 SET, 919
REGION_ROUTINES, 604
 GLOBAL_REGION, 608
 REGION_COORDINATE_SYSTEM_GET,
 604
 REGION_COORDINATE_SYSTEM_SET_-
 NUMBER, 605
 REGION_COORDINATE_SYSTEM_SET_-
 PTR, 605
 REGION_CREATE_FINISH, 605
 REGION_CREATE_START, 605

REGION_DESTROY, 606
 REGION_LABEL_GET, 606
 REGION_LABEL_SET_NUMBER, 606
 REGION_LABEL_SET_PTR, 606
 REGION_SUB_REGION_CREATE_FINISH, 607
 REGION_SUB_REGION_CREATE_START, 607
 REGION_USER_NUMBER_FIND, 607
 REGION_USER_NUMBER_FIND_PTR, 607
 REGIONS_FINALISE, 608
 REGIONS_INITIALISE, 608
 REGION_ROUTINES::REGION_COORDINATE_SYSTEM_SET, 918
 REGION_COORDINATE_SYSTEM_SET_NUMBER, 918
 REGION_COORDINATE_SYSTEM_SET_PTR, 918
 REGION_ROUTINES::REGION_LABEL_SET, 919
 REGION_LABEL_SET_NUMBER, 919
 REGION_LABEL_SET_PTR, 919
 REGION_SUB_REGION_CREATE_FINISH, 607
 REGION_ROUTINES, 607
 REGION_SUB_REGION_CREATE_START, 607
 REGION_ROUTINES, 607
 REGION_USER_NUMBER_FIND, 607
 REGION_ROUTINES, 607
 REGION_USER_NUMBER_FIND_PTR, 607
 REGION_ROUTINES, 607
 REGIONS_FINALISE, 608
 REGIONS_INITIALISE, 608
 REGULAR_MESH TYPES::GENERATED_MESH_TYPE, 1122
 RELATIVE_TOLERANCE TYPES::LINEAR_ITERATIVE_SOLVER_TYPE, 1126
 remove_CH ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::remove, 837
 remove_VS ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::remove, 837
 repeat_ ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::repeat, 838
 replace_CH_CH_auto ISO_VARYING_STRING, 447
 ISO_VARYING_STRING::replace, 839
 replace_CH_CH_CH_target ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 839
 replace_CH_CH_fixed ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 839
 replace_CH_CH_VS_target ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 839
 replace_CH_VS_auto ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 840
 replace_CH_VS_CH_target ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 840
 replace_CH_VS_fixed ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 840
 replace_CH_VS_VS_target ISO_VARYING_STRING, 448
 ISO_VARYING_STRING::replace, 840
 replace_VS_CH_auto ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 840
 replace_VS_CH_CH_target ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 840
 replace_VS_CH_fixed ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 840
 replace_VS_CH_VS_target ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 841
 replace_VS_VS_auto ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 841
 replace_VS_VS_CH_target ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 841
 replace_VS_VS_fixed ISO_VARYING_STRING, 449
 ISO_VARYING_STRING::replace, 841
 replace_VS_VS_VS_target ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::replace, 841
 RESET_BINARY_NUMBER_TAGS BINARY_FILE, 183
 RHS_VARIABLE TYPES::EQUATIONS_MAPPING_TYPE, 1041
 RHS_VARIABLE_MAPPING TYPES::EQUATIONS_MAPPING_TYPE, 1041
 RHS_VARIABLE_TYPE TYPES::EQUATIONS_MAPPING_TYPE, 1042
 RHS_VECTOR

TYPES::SOLVER_MATRICES_TYPE, 1189
RIGHT
 TREES::TREE_NODE_TYPE, 943
ROOT
 TREES::TREE_TYPE, 944
ROUTINE_LIST_ITEM
 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695
ROUTINE_STACK
 BASE_ROUTINES, 174
ROW_DOFS
 TYPES::ELEMENT_MATRIX_TYPE, 1029
 TYPES::ELEMENT_VECTOR_TYPE, 1030
ROW_DOFS_MAPPING
 TYPES::EQUATIONS_MAPPING_TYPE,
 1042
 TYPES::SOLUTION_MAPPING_TYPE,
 1175
ROW_DOMAIN_MAPPING
 TYPES::DISTRIBUTED_MATRIX_TYPE,
 980
ROW_INDICES
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 978
 TYPES::MATRIX_TYPE, 1132
ROW_TO_DOFS_MAP
 TYPES::EQUATIONS_ROW_TO_-
 VARIABLE_MAP_TYPE, 1053
ROW_TO_RHS_DOF
 TYPES::EQUATIONS_ROW_TO_-
 VARIABLE_MAP_TYPE, 1053

SAME_HEADER
 FIELD_IO_ROUTINES::FIELD_IO_INFO_-
 SET, 763
SAMEENDIAN
 binary_file_c.c, 1220
SASUM
 BLAS::interface, 714
SAXPY
 BLAS::interface, 714
SCALE_FACTORS
 TYPES::FIELD_SCALING_TYPE, 1105
SCALING_INDEX
 TYPES::FIELD_VARIABLE_-
 COMPONENT_TYPE, 1114
SCALING_TYPE
 TYPES::FIELD_SCALINGS_TYPE, 1107
SCALINGS
 TYPES::FIELD_SCALINGS_TYPE, 1107
 TYPES::FIELD_TYPE, 1111
scan_CH_VS
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::scan, 842
scan_VS_CH
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::scan, 842
scan_VS_VS
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::scan, 842
SCHEMES
 TYPES::QUADRATURE_TYPE, 1168
SCOPY
 BLAS::interface, 714
SDOT
 BLAS::interface, 715
SEND_BUFFER_DP
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 989
SEND_BUFFER_INTG
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 990
SEND_BUFFER_L
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 990
SEND_BUFFER_SIZE
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 990
SEND_BUFFER_SP
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 990
SEND_TAG_NUMBER
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 990
SET_BINARY_FILE
 BINARY_FILE, 183
SET_INDEX
 TYPES::FIELD_PARAMETER_SET_TYPE,
 1100
SET_TYPE
 TYPES::FIELD_PARAMETER_SET_TYPE,
 1100
 TYPES::FIELD_PARAMETER_SETS_-
 TYPE, 1103
SHAPE_SIZE
 FIELD_IO_ROUTINES, 331
SHELL_SORT_DP
 SORTING, 643
 SORTING::SHELL_SORT, 922
SHELL_SORT_INTG
 SORTING, 643
 SORTING::SHELL_SORT, 922
SHELL_SORT_SP
 SORTING, 644
 SORTING::SHELL_SORT, 922
SHORT_INTEGER_SIZE
 MACHINE_CONSTANTS, 482
SHORTINTTYPE

binary_file_c.c, [1220](#)
 SINGLE_COMPLEX_SIZE
 MACHINE_CONSTANTS, [482](#)
 SINGLE_REAL_SIZE
 MACHINE_CONSTANTS, [482](#)
 SINTEGER_SIZE
 BINARY_FILE::BINARY_FILE_INFO_TYPE, [700](#)
 SINTG
 KINDS_IntegerKinds, [79](#)
 SIZE
 COMP_ENVIRONMENT::CACHE_TYPE, [734](#)
 LISTS::LIST_TYPE, [868](#)
 TYPES::MATRIX_TYPE, [1132](#)
 TYPES::VECTOR_TYPE, [1207](#)
 SKIP_BINARY_FILE
 BINARY_FILE, [183](#)
 SKIP_BINARY_TAGS
 BINARY_FILE, [183](#)
 SKIP_CM_BINARY_HEADER
 BINARY_FILE, [183](#)
 SNESCreate
 CMISS_PETSC::interface, [723](#)
 SNESDestroy
 CMISS_PETSC::interface, [723](#)
 SNESSetFromOptions
 CMISS_PETSC::interface, [723](#)
 SNESSetFunction
 CMISS_PETSC::interface, [723](#)
 SNESSetType
 CMISS_PETSC::interface, [723](#)
 SNESSolve
 CMISS_PETSC::interface, [723](#)
 SNRM2
 BLAS::interface, [715](#)
 SOLUTION
 TYPES::PROBLEM_LINEAR_DATA_TYPE, [1156](#)
 TYPES::PROBLEM_NONLINEAR_DATA_TYPE, [1157](#)
 TYPES::PROBLEM_TIME_DATA_TYPE, [1159](#)
 TYPES::SOLUTION_MAPPING_TYPE, [1176](#)
 TYPES::SOLVER_TYPE, [1197](#)
 SOLUTION_FINISHED
 TYPES::SOLUTION_TYPE, [1179](#)
 SOLUTION_MAPPING
 TYPES::EQUATIONS_MATRICES_TYPE, [1044](#)
 TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE, [1063](#)
 TYPES::SOLUTION_TYPE, [1179](#)
 TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAPS
 SETS_MAP_TYPE, [1184](#)
 SOLUTION_MAPPING_FINISHED
 TYPES::SOLUTION_MAPPING_TYPE, [1176](#)
 SOLUTION_METHOD
 TYPES::EQUATIONS_SET_TYPE, [1066](#)
 SOLUTION_NUMBER
 TYPES::SOLUTION_TYPE, [1179](#)
 SOLUTIONS
 TYPES::PROBLEM_TYPE, [1161](#)
 SOLVE_SMALL_LINEAR_SYSTEM_DP
 MATHS, [490](#)
 MATHS::SOLVE_SMALL_LINEAR_SYSTEM, [883](#)
 SOLVE_SMALL_LINEAR_SYSTEM_SP
 MATHS, [490](#)
 MATHS::SOLVE_SMALL_LINEAR_SYSTEM, [883](#)
 SOLVE_TYPE
 TYPES::SOLVER_TYPE, [1197](#)
 SOLVER
 TYPES::EIGENPROBLEM_SOLVER_TYPE, [1027](#)
 TYPES::LINEAR_SOLVER_TYPE, [1127](#)
 TYPES::NONLINEAR_SOLVER_TYPE, [1154](#)
 TYPES::SOLUTION_TYPE, [1179](#)
 TYPES::SOLVER_MATRICES_TYPE, [1189](#)
 TYPES::TIME_INTEGRATION_SOLVER_TYPE, [1200](#)
 SOLVER_4TH_RUNGE_KUTTA_INTEGRATOR
 SOLVER_ROUTINES, [641](#)
 SOLVER_ADAMS_MOULTON_INTEGRATOR
 SOLVER_ROUTINES, [641](#)
 SOLVER_CMISS_LIBRARY
 SOLVER_ROUTINES_SolverLibraries, [114](#)
 SOLVER_COL_TO_EQUATIONS_MAPS
 TYPES::SOLVER_COL_TO_EQUATIONS_SET_MAP_TYPE, [1182](#)
 SOLVER_COL_TO_EQUATIONS_SET_MAPS
 TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE, [1184](#)
 SOLVER_COL_TO_EQUATIONS_SETS_MAP
 TYPES::SOLUTION_MAPPING_TYPE, [1176](#)
 SOLVER_COL_TO_VARIABLE_MAPS
 TYPES::SOLVER_COL_TO_EQUATIONS_SETS_MAP_TYPE, [1184](#)
 SOLVER_COLS
 TYPES::EQUATIONS_COL_TO_SOLVER_COLS_MAP_TYPE, [1031](#)

SOLVER_CREATE_FINISH
 SOLVER_ROUTINES, 621

SOLVER_CREATE_START
 SOLVER_ROUTINES, 621

SOLVER_DESTROY
 SOLVER_ROUTINES, 622

SOLVER_DIRECT_CHOLESKY
 SOLVER_ROUTINES_-
 DirectLinearSolverTypes, 118

SOLVER_DIRECT_LU
 SOLVER_ROUTINES_-
 DirectLinearSolverTypes, 118

SOLVER_DIRECT_SVD
 SOLVER_ROUTINES_-
 DirectLinearSolverTypes, 118

SOLVER_EIGENPROBLEM_CREATE_FINISH
 SOLVER_ROUTINES, 622

SOLVER_EIGENPROBLEM_FINALISE
 SOLVER_ROUTINES, 623

SOLVER_EIGENPROBLEM_INITIALISE
 SOLVER_ROUTINES, 623

SOLVER_EIGENPROBLEM_SOLVE
 SOLVER_ROUTINES, 623

SOLVER_EIGENPROBLEM_TYPE
 SOLVER_ROUTINES_SolverTypes, 112

SOLVER_EULER_INTEGRATOR
 SOLVER_ROUTINES, 641

SOLVER_FINALISE
 SOLVER_ROUTINES, 624

SOLVER_FINISHED
 TYPES::SOLVER_TYPE, 1198

SOLVER_FULL_MATRICES
 SOLVER_ROUTINES_SparsityTypes, 128

SOLVER_IMPROVED_EULER_INTEGRATOR
 SOLVER_ROUTINES, 641

SOLVER_INITIALISE
 SOLVER_ROUTINES, 624

SOLVER_ITERATIVE_ADDITIVE_-
 SCHWARZ_PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 123

SOLVER_ITERATIVE_BiCGSTAB
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 120

SOLVER_ITERATIVE_BICONJUGATE_-
 GRADIENT
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 121

SOLVER_ITERATIVE_BLOCK_JACOBI_-
 PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 124

SOLVER_ITERATIVE_CHEBYCHEV

SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 121

SOLVER_ITERATIVE_CONJGRAD_SQUARED
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 121

SOLVER_ITERATIVE_CONJUGATE_-
 GRADIENT
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 121

SOLVER_ITERATIVE_GMRES
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 122

SOLVER_ITERATIVE_INCOMPLETE_-
 CHOLESKY_PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 124

SOLVER_ITERATIVE_INCOMPLETE_LU_-
 PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 124

SOLVER_ITERATIVE_JACOBI_-
 PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 124

SOLVER_ITERATIVE_NO_PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 125

SOLVER_ITERATIVE_RICHARDSON
 SOLVER_ROUTINES_-
 IterativeLinearSolverTypes, 122

SOLVER_ITERATIVE_SOR_-
 PRECONDITIONER
 SOLVER_ROUTINES_-
 IterativePreconditionerTypes, 125

SOLVER_LIBRARY
 TYPES::EIGENPROBLEM_SOLVER_-
 TYPE, 1027

 TYPES::LINEAR_DIRECT_SOLVER_-
 TYPE, 1123

 TYPES::LINEAR_ITERATIVE_SOLVER_-
 TYPE, 1126

 TYPES::NONLINEAR_SOLVER_TYPE,
 1154

 TYPES::TIME_INTEGRATION_SOLVER_-
 TYPE, 1200

SOLVER_LIBRARY_SET
 SOLVER_ROUTINES, 625

SOLVER_LINEAR_CREATE_FINISH
 SOLVER_ROUTINES, 625

SOLVER_LINEAR_DIRECT_CREATE_FINISH
 SOLVER_ROUTINES, 626

SOLVER_LINEAR_DIRECT_FINALISE
 SOLVER_ROUTINES, 626

SOLVER_LINEAR_DIRECT_INITIALISE

SOLVER_ROUTINES, 627
 SOLVER_LINEAR_DIRECT_SOLVE
 SOLVER_ROUTINES, 627
 SOLVER_LINEAR_DIRECT_SOLVE_TYPE
 SOLVER_ROUTINES_LinearSolverTypes,
 116
 SOLVER_LINEAR_DIRECT_TYPE_SET
 SOLVER_ROUTINES, 628
 SOLVER_LINEAR_FINALISE
 SOLVER_ROUTINES, 628
 SOLVER_LINEAR_INITIALISE
 SOLVER_ROUTINES, 629
 SOLVER_LINEAR_ITERATIVE_ABSOLUTE_-
 TOLERANCE_SET
 SOLVER_ROUTINES, 629
 SOLVER_LINEAR_ITERATIVE_CREATE_-
 FINISH
 SOLVER_ROUTINES, 629
 SOLVER_LINEAR_ITERATIVE_-
 DIVERGENCE_TOLERANCE_SET
 SOLVER_ROUTINES, 630
 SOLVER_LINEAR_ITERATIVE_FINALISE
 SOLVER_ROUTINES, 631
 SOLVER_LINEAR_ITERATIVE_INITIALISE
 SOLVER_ROUTINES, 631
 SOLVER_LINEAR_ITERATIVE_MAXIMUM_-
 ITERATIONS_SET
 SOLVER_ROUTINES, 631
 SOLVER_LINEAR_ITERATIVE_-
 PRECONDITIONER_TYPE_SET
 SOLVER_ROUTINES, 632
 SOLVER_LINEAR_ITERATIVE_RELATIVE_-
 TOLERANCE_SET
 SOLVER_ROUTINES, 632
 SOLVER_LINEAR_ITERATIVE_SOLVE
 SOLVER_ROUTINES, 633
 SOLVER_LINEAR_ITERATIVE_SOLVE_TYPE
 SOLVER_ROUTINES_LinearSolverTypes,
 116
 SOLVER_LINEAR_ITERATIVE_TYPE_SET
 SOLVER_ROUTINES, 633
 SOLVER_LINEAR_SOLVE
 SOLVER_ROUTINES, 634
 SOLVER_LINEAR_TYPE
 SOLVER_ROUTINES_SolverTypes, 112
 SOLVER_LINEAR_TYPE_SET
 SOLVER_ROUTINES, 634
 SOLVER_LSODA_INTEGRATOR
 SOLVER_ROUTINES, 641
 SOLVER_MATRICES
 TYPES::SOLVER_MATRIX_TYPE, 1193
 TYPES::SOLVER_TYPE, 1198
 SOLVER_MATRICES_ASSEMBLE
 SOLVER_ROUTINES, 635
 SOLVER_MATRICES_CREATE_FINISH
 SOLVER_MATRICES_ROUTINES, 610
 SOLVER_MATRICES_CREATE_START
 SOLVER_MATRICES_ROUTINES, 610
 SOLVER_MATRICES_DESTROY
 SOLVER_MATRICES_ROUTINES, 610
 SOLVER_MATRICES_FINALISE
 SOLVER_MATRICES_ROUTINES, 611
 SOLVER_MATRICES_FINISHED
 TYPES::SOLVER_MATRICES_TYPE, 1189
 SOLVER_MATRICES_INITIALISE
 SOLVER_MATRICES_ROUTINES, 611
 SOLVER_MATRICES_LIBRARY_TYPE_SET
 SOLVER_MATRICES_ROUTINES, 612
 SOLVER_MATRICES_OUTPUT
 SOLVER_MATRICES_ROUTINES, 612
 SOLVER_MATRICES_ROUTINES, 609
 SOLVER_MATRICES_CREATE_FINISH,
 610
 SOLVER_MATRICES_CREATE_START,
 610
 SOLVER_MATRICES_DESTROY, 610
 SOLVER_MATRICES_FINALISE, 611
 SOLVER_MATRICES_INITIALISE, 611
 SOLVER_MATRICES_LIBRARY_TYPE_-
 SET, 612
 SOLVER_MATRICES_OUTPUT, 612
 SOLVER_MATRICES_STORAGE_TYPE_-
 SET, 613
 SOLVER_MATRIX_FINALISE, 613
 SOLVER_MATRIX_INITIALISE, 614
 SOLVER_MATRIX_STRUCTURE_-
 CALCULATE, 614
 SOLVER_MATRICES_STORAGE_TYPE_SET
 SOLVER_MATRICES_ROUTINES, 613
 SOLVER_MATRIX
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_TYPE, 1072
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1184
 SOLVER_MATRIX_FINALISE
 SOLVER_MATRICES_ROUTINES, 613
 SOLVER_MATRIX_INITIALISE
 SOLVER_MATRICES_ROUTINES, 614
 SOLVER_MATRIX_NUMBER
 TYPES::EQUATIONS_TO_SOLVER_-
 MAPS_TYPE, 1072
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1184
 SOLVER_MATRIX_OUTPUT
 SOLVER_ROUTINES_OutputTypes, 126

SOLVER_MATRIX_STRUCTURE_-
 CALCULATE
 SOLVER_MATRICES_ROUTINES, 614
SOLVER_NO_OUTPUT
 SOLVER_ROUTINES_OutputTypes, 126
SOLVER_NONLINEAR_CREATE_FINISH
 SOLVER_ROUTINES, 635
SOLVER_NONLINEAR_FINALISE
 SOLVER_ROUTINES, 636
SOLVER_NONLINEAR_INITIALISE
 SOLVER_ROUTINES, 636
SOLVER_NONLINEAR_SOLVE
 SOLVER_ROUTINES, 636
SOLVER_NONLINEAR_TYPE
 SOLVER_ROUTINES_SolverTypes, 113
SOLVER_OUTPUT_TYPE_SET
 SOLVER_ROUTINES, 637
SOLVER_PETSC_LIBRARY
 SOLVER_ROUTINES_SolverLibraries, 114
SOLVER_ROUTINES, 616
 SOLVER_4TH_RUNGE_KUTTA_-
 INTEGRATOR, 641
 SOLVER_ADAMS_MOULTON_-
 INTEGRATOR, 641
 SOLVER_CREATE_FINISH, 621
 SOLVER_CREATE_START, 621
 SOLVER_DESTROY, 622
 SOLVER_EIGENPROBLEM_CREATE_-
 FINISH, 622
 SOLVER_EIGENPROBLEM_FINALISE,
 623
 SOLVER_EIGENPROBLEM_INITIALISE,
 623
 SOLVER_EIGENPROBLEM_SOLVE, 623
 SOLVER_EULER_INTEGRATOR, 641
 SOLVER_FINALISE, 624
 SOLVER_IMPROVED_EULER_-
 INTEGRATOR, 641
 SOLVER_INITIALISE, 624
 SOLVER_LIBRARY_SET, 625
 SOLVER_LINEAR_CREATE_FINISH, 625
 SOLVER_LINEAR_DIRECT_CREATE_-
 FINISH, 626
 SOLVER_LINEAR_DIRECT_FINALISE,
 626
 SOLVER_LINEAR_DIRECT_INITIALISE,
 627
 SOLVER_LINEAR_DIRECT_SOLVE, 627
 SOLVER_LINEAR_DIRECT_TYPE_SET,
 628
 SOLVER_LINEAR_FINALISE, 628
 SOLVER_LINEAR_INITIALISE, 629
 SOLVER_LINEAR_ITERATIVE_-
 ABSOLUTE_TOLERANCE_SET,
 629
SOLVER_LINEAR_ITERATIVE_CREATE_-
 FINISH, 629
SOLVER_LINEAR_ITERATIVE_-
 DIVERGENCE_TOLERANCE_SET,
 630
SOLVER_LINEAR_ITERATIVE_FINALISE,
 631
SOLVER_LINEAR_ITERATIVE_-
 INITIALISE, 631
SOLVER_LINEAR_ITERATIVE_-
 MAXIMUM_ITERATIONS_SET,
 631
SOLVER_LINEAR_ITERATIVE_-
 PRECONDITIONER_TYPE_SET,
 632
SOLVER_LINEAR_ITERATIVE_-
 RELATIVE_TOLERANCE_SET,
 632
SOLVER_LINEAR_ITERATIVE_SOLVE,
 633
SOLVER_LINEAR_ITERATIVE_TYPE_-
 SET, 633
SOLVER_LINEAR_SOLVE, 634
SOLVER_LINEAR_TYPE_SET, 634
SOLVER_LSODA_INTEGRATOR, 641
SOLVER_MATRICES_ASSEMBLE, 635
SOLVER_NONLINEAR_CREATE_FINISH,
 635
SOLVER_NONLINEAR_FINALISE, 636
SOLVER_NONLINEAR_INITIALISE, 636
SOLVER_NONLINEAR_SOLVE, 636
SOLVER_OUTPUT_TYPE_SET, 637
SOLVER_SOLVE, 637
SOLVER_SPARSITY_TYPE_SET, 638
SOLVER_TIME_INTEGRATION_-
 CREATE_FINISH, 638
SOLVER_TIME_INTEGRATION_-
 FINALISE, 639
SOLVER_TIME_INTEGRATION_-
 INITIALISE, 639
SOLVER_TIME_INTEGRATION_SOLVE,
 640
SOLVER_VARIABLES_UPDATE, 640
SOLVER_ROUTINES::DirectLinearSolverTypes,
 118
SOLVER_ROUTINES::IterativeLinearSolverTypes,
 120
SOLVER_ROUTINES::IterativePreconditionerTypes,
 123
SOLVER_ROUTINES::LinearSolverTypes, 116
SOLVER_ROUTINES::OutputTypes, 126
SOLVER_ROUTINES::SolverLibraries, 114
SOLVER_ROUTINES::SolverTypes, 112

SOLVER_ROUTINES::SparsityTypes, 128
SOLVER_ROUTINES_DirectLinearSolverTypes
 SOLVER_DIRECT_CHOLESKY, 118
 SOLVER_DIRECT_LU, 118
 SOLVER_DIRECT_SVD, 118
SOLVER_ROUTINES_IterativeLinearSolverTypes
 SOLVER_ITERATIVE_BICGSTAB, 120
 SOLVER_ITERATIVE_BICONJUGATE_-
 GRADIENT, 121
 SOLVER_ITERATIVE_CHEBYCHEV, 121
 SOLVER_ITERATIVE_CONJGRAD_-
 SQUARED, 121
 SOLVER_ITERATIVE_CONJUGATE_-
 GRADIENT, 121
 SOLVER_ITERATIVE_GMRES, 122
 SOLVER_ITERATIVE_RICHARDSON, 122
SOLVER_ROUTINES_-
 IterativePreconditionerTypes
 SOLVER_ITERATIVE_ADDITIVE_-
 SCHWARZ_PRECONDITIONER,
 123
 SOLVER_ITERATIVE_BLOCK_JACOBI_-
 PRECONDITIONER, 124
 SOLVER_ITERATIVE_INCOMPLETE_-
 CHOLESKY_PRECONDITIONER,
 124
 SOLVER_ITERATIVE_INCOMPLETE_-
 LU_PRECONDITIONER, 124
 SOLVER_ITERATIVE_JACOBI_-
 PRECONDITIONER, 124
 SOLVER_ITERATIVE_NO_-
 PRECONDITIONER, 125
 SOLVER_ITERATIVE_SOR_-
 PRECONDITIONER, 125
SOLVER_ROUTINES_LinearSolverTypes
 SOLVER_LINEAR_DIRECT_SOLVE_-
 TYPE, 116
 SOLVER_LINEAR_ITERATIVE_SOLVE_-
 TYPE, 116
SOLVER_ROUTINES_OutputTypes
 SOLVER_MATRIX_OUTPUT, 126
 SOLVER_NO_OUTPUT, 126
 SOLVER_SOLVER_OUTPUT, 127
 SOLVER_TIMING_OUTPUT, 127
SOLVER_ROUTINES_SolverLibraries
 SOLVER_CMISS_LIBRARY, 114
 SOLVER_PETSC_LIBRARY, 114
SOLVER_ROUTINES_SolverTypes
 SOLVER_EIGENPROBLEM_TYPE, 112
 SOLVER_LINEAR_TYPE, 112
 SOLVER_NONLINEAR_TYPE, 113
 SOLVER_TIME_INTEGRATION_TYPE,
 113
SOLVER_ROUTINES_SparsityTypes
 SOLVER_FULL_MATRICES, 128
 SOLVER_SPARSE_MATRICES, 128
SOLVER_ROW_TO_EQUATIONS_SET_MAPS
 TYPES::SOLUTION_MAPPING_TYPE,
 1176
SOLVER_ROWS
 TYPES::EQUATIONS_ROW_TO_-
 SOLVER_ROWS_MAP_TYPE, 1052
SOLVER_SOLVE
 SOLVER_ROUTINES, 637
SOLVER_SOLVER_OUTPUT
 SOLVER_ROUTINES_OutputTypes, 127
SOLVER_SPARSE_MATRICES
 SOLVER_ROUTINES_SparsityTypes, 128
SOLVER_SPARSITY_TYPE_SET
 SOLVER_ROUTINES, 638
SOLVER_TIME_INTEGRATION_CREATE_-
 FINISH
 SOLVER_ROUTINES, 638
SOLVER_TIME_INTEGRATION_FINALISE
 SOLVER_ROUTINES, 639
SOLVER_TIME_INTEGRATION_INITIALISE
 SOLVER_ROUTINES, 639
SOLVER_TIME_INTEGRATION_SOLVE
 SOLVER_ROUTINES, 640
SOLVER_TIME_INTEGRATION_TYPE
 SOLVER_ROUTINES_SolverTypes, 113
SOLVER_TIMING_OUTPUT
 SOLVER_ROUTINES_OutputTypes, 127
SOLVER_VARIABLES_UPDATE
 SOLVER_ROUTINES, 640
SOLVER_VECTOR
 TYPES::SOLVER_MATRIX_TYPE, 1193
SORT_METHOD
 LISTS::LIST_TYPE, 868
SORT_ORDER
 LISTS::LIST_TYPE, 869
SORTING, 642
 BUBBLE_SORT_DP, 642
 BUBBLE_SORT_INTG, 642
 BUBBLE_SORT_SP, 642
 HEAP_SORT_DP, 643
 HEAP_SORT_INTG, 643
 HEAP_SORT_SP, 643
 SHELL_SORT_DP, 643
 SHELL_SORT_INTG, 643
 SHELL_SORT_SP, 644
SORTING::BUBBLE_SORT, 920
 BUBBLE_SORT_DP, 920
 BUBBLE_SORT_INTG, 920
 BUBBLE_SORT_SP, 920
SORTING::HEAP_SORT, 921
 HEAP_SORT_DP, 921
 HEAP_SORT_INTG, 921

HEAP_SORT_SP, 921
SORTING::SHELL_SORT, 922
SHELL_SORT_DP, 922
SHELL_SORT_INTG, 922
SHELL_SORT_SP, 922
SOURCE
 TYPES::EQUATIONS_SET_TYPE, 1067
SOURCE_DOF_TO_EQUATIONS_ROW_MAP
 TYPES::SOURCE_EQUATIONS_-
 MATRICES_MAP_TYPE, 1199
SOURCE_FIELD
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
 TYPES::EQUATIONS_SET_SOURCE_-
 TYPE, 1061
SOURCE_FINISHED
 TYPES::EQUATIONS_SET_SOURCE_-
 TYPE, 1061
SOURCE_INTERP_PARAMETERS
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
SOURCE_INTERP_POINT
 TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1035
SOURCE_MAPPINGS
 TYPES::EQUATIONS_MAPPING_TYPE,
 1042
SP
 KINDS_RealKinds, 80
SP_FORMAT
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 700
SP_REAL_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 700
SPARSITY_TYPE
 TYPES::EQUATIONS_TYPE, 1079
 TYPES::SOLVER_TYPE, 1198
SPC
 KINDS_ComplexKinds, 83
SPC_REAL_SIZE
 BINARY_FILE::BINARY_FILE_INFO_-
 TYPE, 700
split_CH
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::split, 843
split_VS
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::split, 843
SROT
 BLAS::interface, 715
SROTG
 BLAS::interface, 715
SSCAL

BLAS::interface, 715
STACK_POINTER
 BASE_ROUTINES::ROUTINE_STACK_-
 TYPE, 697
START_READ_COMFILE_UNIT
 BASE_ROUTINES_FileUnits, 24
STOP_READ_COMFILE_UNIT
 BASE_ROUTINES_FileUnits, 25
STORAGE_TYPE
 TYPES::DISTRIBUTED_MATRIX_-
 PETSC_TYPE, 978
 TYPES::EQUATIONS_MATRIX_TYPE,
 1050
 TYPES::MATRIX_TYPE, 1132
 TYPES::SOLVER_MATRIX_TYPE, 1193
STRING_TO_DOUBLE_C
 STRINGS, 660
 STRINGS::STRING_TO_DOUBLE, 934
STRING_TO_DOUBLE_VS
 STRINGS, 660
 STRINGS::STRING_TO_DOUBLE, 934
STRING_TO_INTEGER_C
 STRINGS, 660
 STRINGS::STRING_TO_INTEGER, 935
STRING_TO_INTEGER_VS
 STRINGS, 661
 STRINGS::STRING_TO_INTEGER, 935
STRING_TO_LOGICAL_C
 STRINGS, 661
 STRINGS::STRING_TO_LOGICAL, 936
STRING_TO_LOGICAL_VS
 STRINGS, 661
 STRINGS::STRING_TO_LOGICAL, 936
STRING_TO_LONG_INTEGER_C
 STRINGS, 662
 STRINGS::STRING_TO_LONG_INTEGER,
 937
STRING_TO_LONG_INTEGER_VS
 STRINGS, 662
 STRINGS::STRING_TO_LONG_INTEGER,
 937
STRING_TO_MUTI_INTEGERS_VS
 FIELD_IO_ROUTINES, 330
STRING_TO_MUTI_REALS_VS
 FIELD_IO_ROUTINES, 330
STRING_TO_SINGLE_C
 STRINGS, 662
 STRINGS::STRING_TO_SINGLE, 938
STRING_TO_SINGLE_VS
 STRINGS, 663
 STRINGS::STRING_TO_SINGLE, 938
STRINGS, 645
 CHARACTER_TO_LOWERCASE_C, 649
 CHARACTER_TO_LOWERCASE_VS, 649

CHARACTER_TO_UPPERCASE_C, 649
 CHARACTER_TO_UPPERCASE_VS, 650
 IS_ABBREVIATION_C_C, 650
 IS_ABBREVIATION_C_VS, 650
 IS_ABBREVIATION_VS_C, 651
 IS_ABBREVIATION_VS_VS, 651
 IS_DIGIT, 651
 IS LETTER, 651
 IS LOWERCASE, 651
 IS_UPPERCASE, 652
 IS_WHITESPACE, 652
 LIST_TO_CHARACTER_C, 652
 LIST_TO_CHARACTER_DP, 653
 LIST_TO_CHARACTER_INTG, 653
 LIST_TO_CHARACTER_L, 654
 LIST_TO_CHARACTER_LINTG, 654
 LIST_TO_CHARACTER_SP, 655
 LOGICAL_TO_CHARACTER, 655
 LOGICAL_TO_VSTRING, 655
 NUMBER_TO_CHARACTER_DP, 656
 NUMBER_TO_CHARACTER_INTG, 656
 NUMBER_TO_CHARACTER_LINTG, 657
 NUMBER_TO_CHARACTER_SP, 657
 NUMBER_TO_VSTRING_DP, 658
 NUMBER_TO_VSTRING_INTG, 658
 NUMBER_TO_VSTRING_LINTG, 659
 NUMBER_TO_VSTRING_SP, 659
 STRING_TO_DOUBLE_C, 660
 STRING_TO_DOUBLE_VS, 660
 STRING_TO_INTEGER_C, 660
 STRING_TO_INTEGER_VS, 661
 STRING_TO_LOGICAL_C, 661
 STRING_TO_LOGICAL_VS, 661
 STRING_TO_LONG_INTEGER_C, 662
 STRING_TO_LONG_INTEGER_VS, 662
 STRING_TO_SINGLE_C, 662
 STRING_TO_SINGLE_VS, 663
 VSTRING_TO_LOWERCASE_C, 663
 VSTRING_TO_LOWERCASE_VS, 663
 VSTRING_TO_UPPERCASE_C, 664
 VSTRING_TO_UPPERCASE_VS, 664
 STRINGS::CHARACTER_TO_LOWERCASE,
 923
 CHARACTER_TO_LOWERCASE_C, 923
 CHARACTER_TO_LOWERCASE_VS, 923
 STRINGS::CHARACTER_TO_UPPERCASE, 924
 CHARACTER_TO_UPPERCASE_C, 924
 CHARACTER_TO_UPPERCASE_VS, 924
 STRINGS::IS_ABBREVIATION, 925
 IS_ABBREVIATION_C_C, 925
 IS_ABBREVIATION_C_VS, 925
 IS_ABBREVIATION_VS_C, 925
 IS_ABBREVIATION_VS_VS, 926
 STRINGS::LIST_TO_CHARACTER, 927
 LIST_TO_CHARACTER_C, 927
 LIST_TO_CHARACTER_DP, 927
 LIST_TO_CHARACTER_INTG, 928
 LIST_TO_CHARACTER_L, 928
 LIST_TO_CHARACTER_LINTG, 928
 LIST_TO_CHARACTER_SP, 929
 STRINGS::NUMBER_TO_CHARACTER, 930
 NUMBER_TO_CHARACTER_DP, 930
 NUMBER_TO_CHARACTER_INTG, 930
 NUMBER_TO_CHARACTER_LINTG, 930
 NUMBER_TO_CHARACTER_SP, 931
 STRINGS::NUMBER_TO_VSTRING, 932
 NUMBER_TO_VSTRING_DP, 932
 NUMBER_TO_VSTRING_INTG, 932
 NUMBER_TO_VSTRING_LINTG, 932
 NUMBER_TO_VSTRING_SP, 933
 STRINGS::STRING_TO_DOUBLE, 934
 STRING_TO_DOUBLE_C, 934
 STRING_TO_DOUBLE_VS, 934
 STRINGS::STRING_TO_INTEGER, 935
 STRING_TO_INTEGER_C, 935
 STRING_TO_INTEGER_VS, 935
 STRINGS::STRING_TO_LOGICAL, 936
 STRING_TO_LOGICAL_C, 936
 STRING_TO_LOGICAL_VS, 936
 STRINGS::STRING_TO_LONG_INTEGER, 937
 STRING_TO_LONG_INTEGER_C, 937
 STRING_TO_LONG_INTEGER_VS, 937
 STRINGS::STRING_TO_SINGLE, 938
 STRING_TO_SINGLE_C, 938
 STRING_TO_SINGLE_VS, 938
 STRINGS::VSTRING_TO_LOWERCASE, 939
 VSTRING_TO_LOWERCASE_C, 939
 VSTRING_TO_LOWERCASE_VS, 939
 STRINGS::VSTRING_TO_UPPERCASE, 940
 VSTRING_TO_UPPERCASE_C, 940
 VSTRING_TO_UPPERCASE_VS, 940
 STRSV
 BLAS::interface, 715
 STRUCTURE_TYPE
 TYPES::EQUATIONS_MATRIX_TYPE,
 1050
 SUB_BASES
 TYPES::BASIS_TYPE, 957
 SUB_REGIONS
 TYPES::REGION_TYPE, 1172
 SUBTYPE
 TYPES::EQUATIONS_SET_TYPE, 1067
 TYPES::PROBLEM_TYPE, 1161
 SURROUNDING_ELEMENTS
 TYPES::DECOMPOSITION_LINE_TYPE,
 966
 TYPES::DOMAIN_NODE_TYPE, 1018
 TYPES::MESH_NODE_TYPE, 1139

SYSTEM_CPU
 TIMER, 665
 timer_c.c, 1380

TEMPORARY_FILE_UNIT
 BASE_ROUTINES_FileUnits, 25

TIME_DATA
 TYPES::EQUATIONS_TYPE, 1079

TIME_INTEGRATION_SOLVER
 TYPES::SOLVER_TYPE, 1198

TIME_TYPE
 TYPES::EQUATIONS_SET_TYPE, 1067

TIMER, 665
 CPU_TIMER, 665
 SYSTEM_CPU, 665
 TOTAL_CPU, 665
 USER_CPU, 665

TIMER::interface, 941
 CPUTIMER, 941

timer_c.c
 CPUTimer, 1380
 SYSTEM_CPU, 1380
 TOTAL_CPU, 1380
 USER_CPU, 1380

TIMING
 BASE_ROUTINES, 174
 BASE_ROUTINES::ROUTINE_STACK_-
 ITEM_TYPE, 695

TIMING_ALL_SUBROUTINES
 BASE_ROUTINES, 174

TIMING_FILE_OPEN
 BASE_ROUTINES, 174

TIMING_FILE_UNIT
 BASE_ROUTINES_FileUnits, 25

TIMING_FROM_SUBROUTINE
 BASE_ROUTINES, 174

TIMING_OUTPUT_TYPE
 BASE_ROUTINES_OutputType, 21

TIMING_ROUTINE_LIST
 BASE_ROUTINES, 175

TIMING_SET_OFF
 BASE_ROUTINES, 168

TIMING_SET_ON
 BASE_ROUTINES, 169

TIMING_SUMMARY
 BASE_ROUTINES, 175

TIMING_SUMMARY_OUTPUT
 BASE_ROUTINES, 169

TOPOLOGY
 TYPES::DECOMPOSITION_TYPE, 972
 TYPES::DOMAIN_TYPE, 1026
 TYPES::MESH_TYPE, 1147

TOTAL_CPU
 TIMER, 665

 timer_c.c, 1380

 TOTAL_EXCLUSIVE_CPU_TIME
 BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 692

 TOTAL_EXCLUSIVE_SYSTEM_TIME
 BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 692

 TOTAL_INCLUSIVE_CPU_TIME
 BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 692

 TOTAL_INCLUSIVE_SYSTEM_TIME
 BASE_ROUTINES::ROUTINE_LIST_-
 ITEM_TYPE, 692

 TOTAL_NUMBER_OF_DOFS
 TYPES::DOMAIN_DOFS_TYPE, 995
 TYPES::FIELD_VARIABLE_TYPE, 1118
 TYPES::SOLVER_COL_TO_EQUATIONS_-
 SETS_MAP_TYPE, 1185

 TOTAL_NUMBER_OF_ELEMENTS
 TYPES::DECOMPOSITION_ELEMENTS_-
 TYPE, 963
 TYPES::DOMAIN_ELEMENTS_TYPE,
 1000

 TOTAL_NUMBER_OF_LOCAL
 TYPES::DOMAIN_MAPPING_TYPE, 1014

 TOTAL_NUMBER_OF_NODES
 TYPES::DOMAIN_NODES_TYPE, 1021

 TOTAL_NUMBER_OF_ROWS
 TYPES::EQUATIONS_MAPPING_TYPE,
 1042
 TYPES::EQUATIONS_MATRICES_TYPE,
 1045
 TYPES::SOLUTION_MAPPING_TYPE,
 1176
 TYPES::SOLVER_MATRICES_TYPE, 1189

TRANSFERS
 TYPES::DISTRIBUTED_VECTOR_-
 CMISS_TYPE, 984

TREE_BLACK_NODE
 TREES_TreeNodeColourTypes, 129

TREE_CREATE_FINISH
 TREES, 669

TREE_CREATE_START
 TREES, 669

TREE_DESTROY
 TREES, 669

TREE_DETACH_AND_DESTROY
 TREES, 670

TREE_DETACH_IN_ORDER
 TREES, 670

TREE_DUPLICATES_ALLOWED_TYPE
 TREES_TreeInsertTypes, 131

TREE_FINALISE
 TREES, 671

TREE_FINISHED
 TREES::TREE_TYPE, 944
 TREE_INITIALISE
 TREES, 671
 TREE_INSERT_TYPE_SET
 TREES, 672
 TREE_ITEM_DELETE
 TREES, 672
 TREE_ITEM_INSERT
 TREES, 673
 TREE_NO_DUPLICATES_ALLOWED
 TREES_TreeInsertTypes, 131
 TREE_NODE_DUPLICATE_KEY
 TREES_TreeNodeInsertStatus, 130
 TREE_NODE_FINALISE
 TREES, 673
 TREE_NODE_INITIALISE
 TREES, 674
 TREE_NODE_INSERT_SUCESSFUL
 TREES_TreeNodeInsertStatus, 130
 TREE_NODE_KEY_GET
 TREES, 674
 TREE_NODE_VALUE_GET
 TREES, 674
 TREE_NODE_VALUE_SET
 TREES, 675
 TREE_OUTPUT
 TREES, 675
 TREE_OUTPUT_IN_ORDER
 TREES, 676
 TREE_PREDECESSOR
 TREES, 676
 TREE_RED_NODE
 TREES_TreeNodeColourTypes, 129
 TREE_SEARCH
 TREES, 677
 TREE_SUCCESSOR
 TREES, 677
 TREES, 667
 TREE_CREATE_FINISH, 669
 TREE_CREATE_START, 669
 TREE_DESTROY, 669
 TREE_DETACH_AND_DESTROY, 670
 TREE_DETACH_IN_ORDER, 670
 TREE_FINALISE, 671
 TREE_INITIALISE, 671
 TREE_INSERT_TYPE_SET, 672
 TREE_ITEM_DELETE, 672
 TREE_ITEM_INSERT, 673
 TREE_NODE_FINALISE, 673
 TREE_NODE_INITIALISE, 674
 TREE_NODE_KEY_GET, 674
 TREE_NODE_VALUE_GET, 674
 TREE_NODE_VALUE_SET, 675
 TREE_OUTPUT, 675
 TREE_OUTPUT_IN_ORDER, 676
 TREE_PREDECESSOR, 676
 TREE_SEARCH, 677
 TREE_SUCCESSOR, 677
 TREES::TREE_NODE_TYPE, 942
 COLOUR, 942
 KEY, 942
 LEFT, 942
 PARENT, 942
 RIGHT, 943
 VALUE, 943
 TREES::TREE_TYPE, 944
 INSERT_TYPE, 944
 NIL, 944
 NUMBER_IN_TREE, 944
 ROOT, 944
 TREE_FINISHED, 944
 TREES::TreeInsertTypes, 131
 TREES::TreeNodeColourTypes, 129
 TREES::TreeNodeInsertStatus, 130
 TREES_TreeInsertTypes
 TREE_DUPLICATES_ALLOWED_TYPE,
 131
 TREE_NO_DUPLICATES_ALLOWED, 131
 TREES_TreeNodeColourTypes
 TREE_BLACK_NODE, 129
 TREE_RED_NODE, 129
 TREES_TreeNodeInsertStatus
 TREE_NODE_DUPLICATE_KEY, 130
 TREE_NODE_INSERT_SUCESSFUL, 130
 trim_
 ISO_VARYING_STRING, 450
 ISO_VARYING_STRING::trim, 844
 TYPE
 TYPES::BASIS_TYPE, 957
 TYPES::COORDINATE_SYSTEM_TYPE,
 959
 TYPES::EQUATIONS_SET_TYPE, 1067
 TYPES::FIELD_TYPE, 1111
 TYPES::PROBLEM_TYPE, 1161
 TYPES::QUADRATURE_TYPE, 1168
 TYPES, 679
 COMP_ENVIRONMENT::MPI_-
 COMPUTATIONAL_NODE_TYPE,
 740
 TYPES::BASIS_FUNCTIONS_TYPE, 946
 BASES, 946
 NUMBER_BASIS_FUNCTIONS, 946
 TYPES::BASIS_PTR_TYPE, 947
 PTR, 947
 TYPES::BASIS_TYPE, 948
 BASIS_FINISHED, 951
 COLLAPSED_XI, 951

DEGENERATE, 951
DERIVATIVE_NUMBERS_IN_LOCAL_-
LINE, 951
DERIVATIVE_ORDER_INDEX, 952
DERIVATIVE_ORDER_INDEX_INV, 952
ELEMENT_PARAMETER_INDEX, 952
FACE_BASES, 952
FAMILY_NUMBER, 952
GLOBAL_NUMBER, 953
HERMITE, 953
INTERPOLATION_ORDER, 953
INTERPOLATION_TYPE, 953
INTERPOLATION_XI, 953
LINE_BASES, 953
LOCAL_LINE_XI_DIRECTION, 954
MAXIMUM_NUMBER_OF_-
DERIVATIVES, 954
NODE_AT_COLLAPSE, 954
NODE_NUMBERS_IN_LOCAL_LINE, 954
NODE_POSITION_INDEX, 954
NODE_POSITION_INDEX_INV, 954
NUMBER_OF_COLLAPSED_XI, 955
NUMBER_OF_DERIVATIVES, 955
NUMBER_OF_ELEMENT_PARAMETERS,
955
NUMBER_OF_LOCAL_LINES, 955
NUMBER_OF_NODES, 955
NUMBER_OF_NODES_IN_LOCAL_LINE,
955
NUMBER_OF_NODES_XI, 955
NUMBER_OF_PARTIAL_DERIVATIVES,
956
NUMBER_OF_SUB_BASES, 956
NUMBER_OF_XI, 956
NUMBER_OF_XI_COORDINATES, 956
PARENT_BASIS, 956
PARTIAL_DERIVATIVE_INDEX, 956
QUADRATURE, 956
SUB_BASES, 957
TYPE, 957
USER_NUMBER, 957
TYPES::COORDINATE_SYSTEM_TYPE, 958
COORDINATE_SYSTEM_FINISHED, 959
FOCUS, 959
NUMBER_OF_DIMENSIONS, 959
ORIENTATION, 959
ORIGIN, 959
RADIAL_INTERPOLATION_TYPE, 959
TYPE, 959
USER_NUMBER, 960
TYPES::DECOMPOSITION_ELEMENT_TYPE,
961
ADJACENT_ELEMENTS, 961
ELEMENT_LINES, 961
GLOBAL_NUMBER, 962
LOCAL_NUMBER, 962
NUMBER_OF_ADJACENT_ELEMENTS,
962
USER_NUMBER, 962
TYPES::DECOMPOSITION_ELEMENTS_-
TYPE, 963
DECOMPOSITION, 963
ELEMENTS, 963
TOTAL_NUMBER_OF_ELEMENTS, 963
TYPES::DECOMPOSITION_LINE_TYPE, 965
ADJACENT_LINES, 965
ELEMENT_LINES, 965
NUMBER, 965
NUMBER_OF_SURROUNDING_-
ELEMENTS, 966
SURROUNDING_ELEMENTS, 966
XI_DIRECTION, 966
TYPES::DECOMPOSITION_LINES_TYPE, 967
DECOMPOSITION, 967
LINES, 967
NUMBER_OF_LINES, 967
TYPES::DECOMPOSITION_PTR_TYPE, 968
PTR, 968
TYPES::DECOMPOSITION_TOPOLOGY_-
TYPE, 969
DECOMPOSITION, 969
ELEMENTS, 969
LINES, 969
TYPES::DECOMPOSITION_TYPE, 970
DECOMPOSITION_FINISHED, 971
DECOMPOSITION_TYPE, 971
DECOMPOSITIONS, 971
DOMAIN, 971
ELEMENT_DOMAIN, 971
GLOBAL_NUMBER, 971
MESH, 972
MESH_COMPONENT_NUMBER, 972
NUMBER_OF_DOMAINS, 972
NUMBER_OF_EDGES_CUT, 972
TOPOLOGY, 972
USER_NUMBER, 972
TYPES::DECOMPOSITIONS_TYPE, 973
DECOMPOSITIONS, 973
MESH, 973
NUMBER_OF_DECOMPOSITIONS, 973
TYPES::DISTRIBUTED_MATRIX_CMISS_-
TYPE, 974
BASE_TAG_NUMBER, 974
DISTRIBUTED_MATRIX, 974
MATRIX, 974
TYPES::DISTRIBUTED_MATRIX_PETSC_-
TYPE, 975
COLUMN_INDICES, 976

DATA_DP, 976
 DATA_SIZE, 976
 DIAGONAL_NUMBER_NON_ZEROS, 976
 DISTRIBUTED_MATRIX, 976
 GLOBAL_M, 977
 GLOBAL_N, 977
 GLOBAL_ROW_NUMBERS, 977
 ISLTGMAPPING, 977
 M, 977
 MATRIX, 977
 MAXIMUM_COLUMN_INDICES_PER_-
 ROW, 977
 N, 977
 NUMBER_NON_ZEROS, 978
 OFFDIAGONAL_NUMBER_NON_ZEROS,
 978
 ROW_INDICES, 978
 STORAGE_TYPE, 978
 TYPES::DISTRIBUTED_MATRIX_TYPE, 979
 CMISS, 979
 COLUMN_DOMAIN_MAPPING, 979
 DATA_TYPE, 980
 GHOSTING_TYPE, 980
 LIBRARY_TYPE, 980
 MATRIX_FINISHED, 980
 PETSC, 980
 ROW_DOMAIN_MAPPING, 980
 TYPES::DISTRIBUTED_VECTOR_CMISS_-
 TYPE, 982
 BASE_TAG_NUMBER, 983
 DATA_DP, 983
 DATA_INTG, 983
 DATA_L, 983
 DATA_SIZE, 983
 DATA_SP, 983
 DISTRIBUTED_VECTOR, 983
 N, 983
 TRANSFERS, 984
 TYPES::DISTRIBUTED_VECTOR_PETSC_-
 TYPE, 985
 DATA_SIZE, 985
 DISTRIBUTED_VECTOR, 985
 GLOBAL_N, 985
 GLOBAL_NUMBERS, 986
 ISLTGMAPPING, 986
 N, 986
 VECTOR, 986
 TYPES::DISTRIBUTED_VECTOR_-
 TRANSFER_TYPE, 987
 CMISS_VECTOR, 988
 DATA_TYPE, 988
 MPI_RECEIVE_REQUEST, 988
 MPI_SEND_REQUEST, 988
 RECEIVE_BUFFER_DP, 989
 RECEIVE_BUFFER_INTG, 989
 RECEIVE_BUFFER_L, 989
 RECEIVE_BUFFER_SIZE, 989
 RECEIVE_BUFFER_SP, 989
 RECEIVE_TAG_NUMBER, 989
 SEND_BUFFER_DP, 989
 SEND_BUFFER_INTG, 990
 SEND_BUFFER_L, 990
 SEND_BUFFER_SIZE, 990
 SEND_BUFFER_SP, 990
 SEND_TAG_NUMBER, 990
 TYPES::DISTRIBUTED_VECTOR_TYPE, 991
 CMISS, 991
 DATA_TYPE, 991
 DOMAIN_MAPPING, 992
 GHOSTING_TYPE, 992
 LIBRARY_TYPE, 992
 PETSC, 992
 VECTOR_FINISHED, 992
 TYPES::DOMAIN_ADJACENT_DOMAIN_-
 TYPE, 993
 DOMAIN_NUMBER, 993
 LOCAL_GHOST_RECEIVE_INDICES, 993
 LOCAL_GHOST_SEND_INDICES, 993
 NUMBER_OF_RECEIVE_GHOSTS, 994
 NUMBER_OF_SEND_GHOSTS, 994
 TYPES::DOMAIN_DOFS_TYPE, 995
 DOF_INDEX, 995
 DOMAIN, 995
 NUMBER_OF_DOFS, 995
 TOTAL_NUMBER_OF_DOFS, 995
 TYPES::DOMAIN_ELEMENT_TYPE, 997
 BASIS, 997
 ELEMENT_DERIVATIVES, 997
 ELEMENT_NODES, 997
 NUMBER, 997
 TYPES::DOMAIN_ELEMENTS_TYPE, 999
 DOMAIN, 999
 ELEMENTS, 999
 MAXIMUM_NUMBER_OF_ELEMENT_-
 PARAMETERS, 999
 NUMBER_OF_ELEMENTS, 1000
 TOTAL_NUMBER_OF_ELEMENTS, 1000
 TYPES::DOMAIN_FACE_PTR_TYPE, 1001
 PTR, 1001
 TYPES::DOMAIN_FACE_TYPE, 1002
 BASIS, 1002
 DERIVATIVES_IN_FACE, 1002
 NODES_IN_FACE, 1002
 NUMBER, 1003
 XI_DIRECTION1, 1003
 XI_DIRECTION2, 1003
 TYPES::DOMAIN_FACES_TYPE, 1004
 DOMAIN, 1004

FACES, 1004
NUMBER_OF_FACES, 1004
TYPES::DOMAIN_GLOBAL_MAPPING_TYPE,
 1005
 DOMAIN_NUMBER, 1005
 LOCAL_NUMBER, 1005
 LOCAL_TYPE, 1005
 NUMBER_OF_DOMAINS, 1006
TYPES::DOMAIN_LINE_PTR_TYPE, 1007
 PTR, 1007
TYPES::DOMAIN_LINE_TYPE, 1008
 BASIS, 1008
 DERIVATIVES_IN_LINE, 1008
 NODES_IN_LINE, 1008
 NUMBER, 1008
TYPES::DOMAIN_LINES_TYPE, 1010
 DOMAIN, 1010
 LINES, 1010
 NUMBER_OF_LINES, 1010
TYPES::DOMAIN_MAPPING_TYPE, 1011
 ADJACENT_DOMAINS, 1012
 ADJACENT_DOMAINS_LIST, 1012
 ADJACENT_DOMAINS_PTR, 1012
 BOUNDARY_LIST, 1013
 GHOST_LIST, 1013
 GLOBAL_TO_LOCAL_MAP, 1013
 INTERNAL_LIST, 1013
 LOCAL_TO_GLOBAL_MAP, 1013
 NUMBER_OF_ADJACENT_DOMAINS,
 1013
 NUMBER_OF_BOUNDARY, 1013
 NUMBER_OF_DOMAIN_LOCAL, 1014
 NUMBER_OF_DOMAINS, 1014
 NUMBER_OF_GHOST, 1014
 NUMBER_OF_GLOBAL, 1014
 NUMBER_OF_INTERNAL, 1014
 NUMBER_OF_LOCAL, 1014
 TOTAL_NUMBER_OF_LOCAL, 1014
TYPES::DOMAIN_MAPPINGS_TYPE, 1015
 DOFS, 1015
 DOMAIN, 1015
 ELEMENTS, 1015
 NODES, 1015
TYPES::DOMAIN_NODE_TYPE, 1017
 DOF_INDEX, 1018
 GLOBAL_NUMBER, 1018
 LOCAL_NUMBER, 1018
 NODE_LINES, 1018
 NUMBER_OF_DERIVATIVES, 1018
 NUMBER_OF_NODE_LINES, 1018
 NUMBER_OF_SURROUNDING_-
 ELEMENTS, 1018
 PARTIAL_DERIVATIVE_INDEX, 1018
 SURROUNDING_ELEMENTS, 1018
USER_NUMBER, 1019
TYPES::DOMAIN_NODES_TYPE, 1020
 DOMAIN, 1020
 MAXIMUM_NUMBER_OF_-
 DERIVATIVES, 1020
 NODES, 1020
 NUMBER_OF_NODES, 1021
 TOTAL_NUMBER_OF_NODES, 1021
TYPES::DOMAIN_PTR_TYPE, 1022
 PTR, 1022
TYPES::DOMAIN_TOPOLOGY_TYPE, 1023
 DOFS, 1023
 DOMAIN, 1023
 ELEMENTS, 1023
 FACES, 1024
 LINES, 1024
 NODES, 1024
TYPES::DOMAIN_TYPE, 1025
 DECOMPOSITION, 1025
 MAPPINGS, 1025
 MESH, 1026
 MESH_COMPONENT_NUMBER, 1026
 NODE_DOMAIN, 1026
 NUMBER_OF_DIMENSIONS, 1026
 REGION, 1026
 TOPOLOGY, 1026
TYPES::EIGENPROBLEM_SOLVER_TYPE,
 1027
 SOLVER, 1027
 SOLVER_LIBRARY, 1027
TYPES::ELEMENT_MATRIX_TYPE, 1028
 COLUMN_DOFS, 1028
 EQUATIONS_MATRIX_NUMBER, 1028
 MATRIX, 1028
 MAX_NUMBER_OF_COLUMNS, 1028
 MAX_NUMBER_OF_ROWS, 1028
 NUMBER_OF_COLUMNS, 1028
 NUMBER_OF_ROWS, 1029
 ROW_DOFS, 1029
TYPES::ELEMENT_VECTOR_TYPE, 1030
 MAX_NUMBER_OF_ROWS, 1030
 NUMBER_OF_ROWS, 1030
 ROW_DOFS, 1030
 VECTOR, 1030
TYPES::EQUATIONS_COL_TO_SOLVER_-
 COLS_MAP_TYPE, 1031
 COUPLING_COEFFICIENTS, 1031
 NUMBER_OF_SOLVER_COLS, 1031
 SOLVER_COLS, 1031
TYPES::EQUATIONS_INTERPOLATION_-
 TYPE, 1032
 DEPENDENT_FIELD, 1033
 DEPENDENT_INTERP_PARAMETERS,
 1033

DEPENDENT_INTERP_POINT, 1033
 EQUATIONS, 1033
 FIBRE_FIELD, 1034
 FIBRE_INTERP_PARAMETERS, 1034
 FIBRE_INTERP_POINT, 1034
 FIBRE_INTERP_POINT_METRICS, 1034
 GEOMETRIC_FIELD, 1034
 GEOMETRIC_INTERP_PARAMETERS,
 1034
 GEOMETRIC_INTERP_POINT, 1034
 GEOMETRIC_INTERP_POINT_METRICS,
 1035
 MATERIAL_FIELD, 1035
 MATERIAL_INTERP_PARAMETERS, 1035
 MATERIAL_INTERP_POINT, 1035
 SOURCE_FIELD, 1035
 SOURCE_INTERP_PARAMETERS, 1035
 SOURCE_INTERP_POINT, 1035
 TYPES::EQUATIONS_LINEAR_DATA_TYPE,
 1037
 EQUATIONS, 1037
 TYPES::EQUATIONS_MAPPING_CREATE_-
 VALUES_CACHE_TYPE, 1038
 MATRIX_COEFFICIENTS, 1038
 MATRIX_VARIABLE_TYPES, 1038
 TYPES::EQUATIONS_MAPPING_TYPE, 1039
 CREATE_VALUES_CACHE, 1040
 EQUATIONS, 1040
 EQUATIONS_MAPPING_FINISHED, 1040
 EQUATIONS_MATRICES, 1040
 EQUATIONS_MATRIX_TO_VARIABLE_-
 MAPS, 1040
 EQUATIONS_ROW_TO_VARIABLE_-
 MAPS, 1041
 MATRIX_VARIABLE_TYPES, 1041
 NUMBER_OF_EQUATIONS_MATRICES,
 1041
 NUMBER_OF_MATRIX_VARIABLES,
 1041
 NUMBER_OF_ROWS, 1041
 RHS_VARIABLE, 1041
 RHS_VARIABLE_MAPPING, 1041
 RHS_VARIABLE_TYPE, 1042
 ROW_DOFS_MAPPING, 1042
 SOURCE_MAPPINGS, 1042
 TOTAL_NUMBER_OF_ROWS, 1042
 VARIABLE_TO_EQUATIONS_-
 MATRICES_MAPS, 1042
 TYPES::EQUATIONS_MATRICES_TYPE, 1043
 ELEMENT_VECTOR, 1044
 EQUATIONS, 1044
 EQUATIONS_MAPPING, 1044
 EQUATIONS_MATRICES_FINISHED, 1044
 MATRICES, 1044
 NUMBER_OF_MATRICES, 1044
 NUMBER_OF_ROWS, 1044
 SOLUTION_MAPPING, 1044
 TOTAL_NUMBER_OF_ROWS, 1045
 UPDATE_VECTOR, 1045
 VECTOR, 1045
 TYPES::EQUATIONS_MATRIX_PTR_TYPE,
 1046
 PTR, 1046
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1047
 COLUMN_DOFS_MAPPING, 1047
 COLUMN_TO_DOF_MAP, 1047
 EQUATIONS_MATRIX, 1048
 MATRIX_COEFFICIENT, 1048
 MATRIX_NUMBER, 1048
 NUMBER_OF_COLUMNS, 1048
 VARIABLE, 1048
 VARIABLE_TYPE, 1048
 TYPES::EQUATIONS_MATRIX_TYPE, 1049
 ELEMENT_MATRIX, 1049
 EQUATIONS_MATRICES, 1049
 MATRIX, 1050
 MATRIX_NUMBER, 1050
 NUMBER_OF_COLUMNS, 1050
 STORAGE_TYPE, 1050
 STRUCTURE_TYPE, 1050
 UPDATE_MATRIX, 1050
 TYPES::EQUATIONS_NONLINEAR_DATA_-
 TYPE, 1051
 EQUATIONS, 1051
 TYPES::EQUATIONS_ROW_TO_SOLVER_-
 ROWS_MAP_TYPE, 1052
 COUPLING_COEFFICIENTS, 1052
 NUMBER_OF_SOLVER_ROWS, 1052
 SOLVER_ROWS, 1052
 TYPES::EQUATIONS_ROW_TO_VARIABLE_-
 MAP_TYPE, 1053
 ROW_TO_DOFS_MAP, 1053
 ROW_TO_RHS_DOF, 1053
 TYPES::EQUATIONS_SET_ANALYTIC_TYPE,
 1054
 ANALYTIC_FINISHED, 1054
 EQUATIONS_SET, 1054
 TYPES::EQUATIONS_SET_DEPENDENT_-
 TYPE, 1055
 DEPENDENT_FIELD, 1055
 DEPENDENT_FINISHED, 1055
 EQUATIONS_SET, 1055
 TYPES::EQUATIONS_SET_FIXED_-
 CONDITIONS_TYPE, 1056
 BOUNDARY_CONDITIONS, 1056
 EQUATIONS_SET, 1056
 FIXED_CONDITIONS_FINISHED, 1056

GLOBAL_BOUNDARY_CONDITIONS, 1056
TYPES::EQUATIONS_SET_GEOMETRY_TYPE, 1058
EQUATIONS_SET, 1058
FIBRE_FIELD, 1058
GEOMETRIC_FIELD, 1058
TYPES::EQUATIONS_SET_MATERIALS_TYPE, 1059
EQUATIONS_SET, 1059
MATERIAL_FIELD, 1059
MATERIALS_FINISHED, 1059
TYPES::EQUATIONS_SET_PTR_TYPE, 1060
PTR, 1060
TYPES::EQUATIONS_SET_SOURCE_TYPE, 1061
EQUATIONS_SET, 1061
SOURCE_FIELD, 1061
SOURCE_FINISHED, 1061
TYPES::EQUATIONS_SET_TO_SOLVER_MAP_TYPE, 1062
EQUATIONS, 1062
EQUATIONS_ROW_TO_SOLVER_ROWS_MAPS, 1062
EQUATIONS_SET_INDEX, 1063
EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM, 1063
EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM, 1063
SOLUTION_MAPPING, 1063
TYPES::EQUATIONS_SET_TYPE, 1064
ANALYTIC, 1065
CLASS, 1065
DEPENDENT, 1065
EQUATIONS, 1065
EQUATIONS_SET_FINISHED, 1065
EQUATIONS_SETS, 1065
FIXED_CONDITIONS, 1066
GEOMETRY, 1066
GLOBAL_NUMBER, 1066
LINEARITY, 1066
MATERIALS, 1066
REGION, 1066
SOLUTION_METHOD, 1066
SOURCE, 1067
SUBTYPE, 1067
TIME_TYPE, 1067
TYPE, 1067
USER_NUMBER, 1067
TYPES::EQUATIONS_SETS_TYPE, 1068
EQUATIONS_SETS, 1068
NUMBER_OF_EQUATIONS_SETS, 1068
REGION, 1068
TYPES::EQUATIONS_TIME_DATA_TYPE, 1069
EQUATIONS, 1069
TYPES::EQUATIONS_TO_SOLVER_MAPS_PTR_TYPE, 1070
PTR, 1070
TYPES::EQUATIONS_TO_SOLVER_MAPS_TYPE, 1071
EQUATIONS_COL_SOLVER_COLS_MAP, 1071
EQUATIONS_MATRIX, 1071
EQUATIONS_MATRIX_NUMBER, 1071
SOLVER_MATRIX, 1072
SOLVER_MATRIX_NUMBER, 1072
TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_EM_TYPE, 1073
EQUATIONS_MATRIX_NUMBER, 1073
EQUATIONS_TO_SOLVER_MATRIX_MAPS, 1073
NUMBER_OF_SOLVER_MATRICES, 1073
TYPES::EQUATIONS_TO_SOLVER_MATRIX_MAPS_SM_TYPE, 1075
EQUATIONS_TO_SOLVER_MATRIX_MAPS, 1075
NUMBER_OF_EQUATIONS_MATRICES, 1076
NUMBER_OF_VARIABLES, 1076
SOLVER_MATRIX_NUMBER, 1076
VARIABLE_TO_SOLVER_COL_MAPS, 1076
VARIABLE_TYPES, 1076
VARIABLES, 1076
TYPES::EQUATIONS_TYPE, 1077
EQUATIONS_FINISHED, 1077
EQUATIONS_MAPPING, 1077
EQUATIONS_MATRICES, 1078
EQUATIONS_SET, 1078
INTERPOLATION, 1078
LINEAR_DATA, 1078
NONLINEAR_DATA, 1078
OUTPUT_TYPE, 1078
SPARSITY_TYPE, 1079
TIME_DATA, 1079
TYPES::FIELD_CREATE_VALUES_CACHE_TYPE, 1080
INTERPOLATION_TYPE, 1080
MESH_COMPONENT_NUMBER, 1080
NUMBER_OF_COMPONENTS, 1080
VARIABLE_TYPES, 1081
TYPES::FIELD_DOF_TO_PARAM_MAP_TYPE, 1082
CONSTANT_DOF2PARAM_MAP, 1083
DOF_TYPE, 1083
ELEMENT_DOF2PARAM_MAP, 1083
NODE_DOF2PARAM_MAP, 1083
NUMBER_OF_CONSTANT_DOFS, 1084

NUMBER_OF_DOFS, 1084
 NUMBER_OF_ELEMENT_DOFS, 1084
 NUMBER_OF_NODE_DOFS, 1084
 NUMBER_OF_POINT_DOFS, 1084
 POINT_DOF2PARAM_MAP, 1084
 VARIABLE_DOF, 1085
TYPES::FIELD_GEOMETRIC_PARAMETERS_TYPE, 1086
 AREAS, 1086
 FIELDS_USING, 1086
 LENGTHS, 1087
 NUMBER_OF_AREAS, 1087
 NUMBER_OF_FIELDS_USING, 1087
 NUMBER_OF_LINES, 1087
 NUMBER_OF_VOLUMES, 1087
 VOLUMES, 1087
TYPES::FIELD_INTERPOLATED_POINT_METRICS_TYPE, 1088
 DX_DXI, 1088
 DXGI_DX, 1088
 GL, 1089
 GU, 1089
 INTERPOLATED_POINT, 1089
 JACOBIAN, 1089
 JACOBIAN_TYPE, 1089
 NUMBER_OF_X_DIMENSIONS, 1089
 NUMBER_OF_XI_DIMENSIONS, 1089
TYPES::FIELD_INTERPOLATED_POINT_TYPE, 1091
 INTERPOLATION_PARAMETERS, 1091
 MAX_PARTIAL_DERIVATIVE_INDEX, 1091
 PARTIAL_DERIVATIVE_TYPE, 1091
 VALUES, 1092
TYPES::FIELD_INTERPOLATION_PARAMETERS_TYPE, 1093
 BASES, 1093
 FIELD, 1093
 FIELD_VARIABLE, 1093
 NUMBER_OF_PARAMETERS, 1094
 PARAMETERS, 1094
TYPES::FIELD_MAPPINGS_TYPE, 1095
 DOF_TO_PARAM_MAP, 1095
 DOMAIN_MAPPING, 1095
TYPES::FIELD_PARAM_TO_DOF_MAP_TYPE, 1096
 CONSTANT_PARAM2DOF_MAP, 1097
 ELEMENT_PARAM2DOF_MAP, 1097
 MAX_NUMBER_OF_DERIVATIVES, 1097
 NODE_PARAM2DOF_MAP, 1097
 NUMBER_OF_CONSTANT_PARAMETERS, 1097
 NUMBER_OF_ELEMENT_PARAMETERS, 1097
 NUMBER_OF_NODE_PARAMETERS, 1098
 NUMBER_OF_POINT_PARAMETERS, 1098
 POINT_PARAM2DOF_MAP, 1098
TYPES::FIELD_PARAMETER_SET_PTR_TYPE, 1099
TYPES::FIELD_PARAMETER_SET_TYPE, 1100
 PARAMETERS, 1100
 SET_INDEX, 1100
 SET_TYPE, 1100
TYPES::FIELD_PARAMETER_SETS_TYPE, 1102
 FIELD, 1102
 NUMBER_OF_PARAMETER_SETS, 1102
 PARAMETER_SETS, 1102
 SET_TYPE, 1103
TYPES::FIELD_PTR_TYPE, 1104
 PTR, 1104
TYPES::FIELD_SCALING_TYPE, 1105
 MAX_NUMBER_OF_DERIVATIVES, 1105
 MAX_NUMBER_OF_ELEMENT_PARAMETERS, 1105
 MESH_COMPONENT_NUMBER, 1105
 SCALE_FACTORS, 1105
TYPES::FIELD_SCALINGS_TYPE, 1107
 NUMBER_OF_SCALING_INDICES, 1107
 SCALING_TYPE, 1107
 SCALINGS, 1107
TYPES::FIELD_TYPE, 1108
 DECOMPOSITION, 1109
 DEPENDENT_TYPE, 1109
 DIMENSION, 1109
 FIELD_FINISHED, 1109
 FIELDS, 1110
 GEOMETRIC_FIELD, 1110
 GEOMETRIC_FIELD_PARAMETERS, 1110
 GLOBAL_NUMBER, 1110
 MAPPINGS, 1110
 NUMBER_OF_VARIABLES, 1110
 PARAMETER_SETS, 1110
 REGION, 1110
 SCALINGS, 1111
 TYPE, 1111
 USER_NUMBER, 1111
 VARIABLE_TYPE_MAP, 1111
 VARIABLES, 1111
TYPES::FIELD_VARIABLE_COMPONENT_TYPE, 1112
 COMPONENT_NUMBER, 1113
 DOMAIN, 1113
 FIELD, 1113
 FIELD_VARIABLE, 1113

INTERPOLATION_TYPE, 1113
MAX_NUMBER_OF_INTERPOLATION_PARAMETERS, 1113
MESH_COMPONENT_NUMBER, 1113
PARAM_TO_DOF_MAP, 1113
REGION, 1114
SCALING_INDEX, 1114
TYPES::FIELD_VARIABLE_PTR_TYPE, 1115
PTR, 1115
TYPES::FIELD_VARIABLE_TYPE, 1116
COMPONENTS, 1117
DOF_LIST, 1117
DOMAIN_MAPPING, 1117
FIELD, 1117
GLOBAL_DOF_LIST, 1117
GLOBAL_DOF_OFFSET, 1117
MAX_NUMBER_OF_INTERPOLATION_PARAMETERS, 1117
NUMBER_OF_COMPONENTS, 1118
NUMBER_OF_DOFS, 1118
REGION, 1118
TOTAL_NUMBER_OF_DOFS, 1118
VARIABLE_NUMBER, 1118
VARIABLE_TYPE, 1118
TYPES::FIELDS_TYPE, 1119
FIELDS, 1119
NUMBER_OF_FIELDS, 1119
REGION, 1119
TYPES::GENERATED_MESH_REGULAR_TYPE, 1120
BASIS, 1120
GENERATED_MESH, 1120
MAXIMUM_EXTENT, 1120
MESH_DIMENSION, 1121
NUMBER_OF_ELEMENTS_XI, 1121
ORIGIN, 1121
TYPES::GENERATED_MESH_TYPE, 1122
GENERATED_TYPE, 1122
MESH, 1122
REGION, 1122
REGULAR_MESH, 1122
USER_NUMBER, 1122
TYPES::LINEAR_DIRECT_SOLVER_TYPE, 1123
DIRECT_SOLVER_TYPE, 1123
LINEAR_SOLVER, 1123
SOLVER_LIBRARY, 1123
TYPES::LINEAR_ITERATIVE_SOLVER_TYPE, 1124
ABSOLUTE_TOLERANCE, 1125
DIVERGENCE_TOLERANCE, 1125
ITERATIVE_PRECONDITIONER_TYPE, 1125
ITERATIVE_SOLVER_TYPE, 1125
KSP, 1125
LINEAR_SOLVER, 1125
MAXIMUM_NUMBER_OF_ITERATIONS, 1125
PC, 1125
RELATIVE_TOLERANCE, 1126
SOLVER_LIBRARY, 1126
TYPES::LINEAR_SOLVER_TYPE, 1127
DIRECT_SOLVER, 1127
ITERATIVE_SOLVER, 1127
LINEAR_SOLVER_TYPE, 1127
SOLVER, 1127
TYPES::MATRIX_TYPE, 1129
COLUMN_INDICES, 1130
DATA_DP, 1130
DATA_INTG, 1130
DATA_L, 1130
DATA_SP, 1130
DATA_TYPE, 1131
ID, 1131
M, 1131
MATRIX_FINISHED, 1131
MAX_M, 1131
MAX_N, 1131
MAXIMUM_COLUMN_INDICES_PER_ROW, 1131
N, 1131
NUMBER_NON_ZEROS, 1132
ROW_INDICES, 1132
SIZE, 1132
STORAGE_TYPE, 1132
TYPES::MESH_DOFS_TYPE, 1133
MESH, 1133
NUMBER_OF_DOFS, 1133
TYPES::MESH_ELEMENT_TYPE, 1134
ADJACENT_ELEMENTS, 1134
BASIS, 1135
GLOBAL_ELEMENT_NODES, 1135
GLOBAL_NUMBER, 1135
NUMBER_OF_ADJACENT_ELEMENTS, 1135
USER_ELEMENT_NODES, 1135
USER_NUMBER, 1135
TYPES::MESH_ELEMENTS_TYPE, 1136
ELEMENTS, 1136
ELEMENTS_FINISHED, 1136
MESH, 1136
NUMBER_OF_ELEMENTS, 1137
TYPES::MESH_NODE_TYPE, 1138
DOF_INDEX, 1138
GLOBAL_NUMBER, 1138
NUMBER_OF_DERIVATIVES, 1138
NUMBER_OF_SURROUNDING_ELEMENTS, 1139

PARTIAL_DERIVATIVE_INDEX, 1139
 SURROUNDING_ELEMENTS, 1139
 USER_NUMBER, 1139
 TYPES::MESH_NODES_TYPE, 1140
 MESH, 1140
 NODES, 1140
 NUMBER_OF_NODES, 1140
 TYPES::MESH_PTR_TYPE, 1141
 PTR, 1141
 TYPES::MESH_TOPOLOGY_PTR_TYPE, 1142
 PTR, 1142
 TYPES::MESH_TOPOLOGY_TYPE, 1143
 DOFS, 1143
 ELEMENTS, 1143
 MESH, 1143
 MESH_COMPONENT_NUMBER, 1143
 NODES, 1144
 TYPES::MESH_TYPE, 1145
 DECOMPOSITIONS, 1146
 EMBEDDED_MESHES, 1146
 EMBEDDING_MESH, 1146
 GLOBAL_NUMBER, 1146
 MESH_EMBEDDED, 1146
 MESH_FINISHED, 1146
 MESHES, 1147
 NUMBER_OF_COMPONENTS, 1147
 NUMBER_OF_DIMENSIONS, 1147
 NUMBER_OF_ELEMENTS, 1147
 NUMBER_OF_EMBEDDED_MESHES,
 1147
 NUMBER_OF_FACES, 1147
 NUMBER_OF_LINES, 1147
 REGION, 1147
 TOPOLOGY, 1147
 USER_NUMBER, 1148
 TYPES::MESHES_TYPE, 1149
 MESHES, 1149
 NUMBER_OF_MESHES, 1149
 REGION, 1149
 TYPES::NODE_TYPE, 1150
 GLOBAL_NUMBER, 1150
 INITIAL_POSITION, 1150
 LABEL, 1150
 USER_NUMBER, 1150
 TYPES::NODES_TYPE, 1152
 NODE_TREE, 1152
 NODES, 1152
 NODES_FINISHED, 1152
 NUMBER_OF_NODES, 1153
 REGION, 1153
 TYPES::NONLINEAR_SOLVER_TYPE, 1154
 SOLVER, 1154
 SOLVER_LIBRARY, 1154
 TYPES::PROBLEM_CONTROL_TYPE, 1155
 CONTROL_FINISHED, 1155
 CONTROL_TYPE, 1155
 PROBLEM, 1155
 TYPES::PROBLEM_LINEAR_DATA_TYPE,
 1156
 SOLUTION, 1156
 TYPES::PROBLEM_NONLINEAR_DATA_-
 TYPE, 1157
 NUMBER_OF_ITERATIONS, 1157
 SOLUTION, 1157
 TYPES::PROBLEM_PTR_TYPE, 1158
 PTR, 1158
 TYPES::PROBLEM_TIME_DATA_TYPE, 1159
 SOLUTION, 1159
 TYPES::PROBLEM_TYPE, 1160
 CLASS, 1160
 CONTROL, 1160
 GLOBAL_NUMBER, 1161
 NUMBER_OF_SOLUTIONS, 1161
 PROBLEM_FINISHED, 1161
 PROBLEMS, 1161
 SOLUTIONS, 1161
 SUBTYPE, 1161
 TYPE, 1161
 USER_NUMBER, 1161
 TYPES::PROBLEMS_TYPE, 1163
 NUMBER_OF_PROBLEMS, 1163
 PROBLEMS, 1163
 TYPES::QUADRATURE_SCHEME_PTR_TYPE,
 1164
 PTR, 1164
 TYPES::QUADRATURE_SCHEME_TYPE, 1165
 GAUSS_BASIS_FNS, 1165
 GAUSS_POSITIONS, 1165
 GAUSS_WEIGHTS, 1166
 GLOBAL_NUMBER, 1166
 NUMBER_OF_GAUSS, 1166
 QUADRATURE, 1166
 TYPES::QUADRATURE_TYPE, 1167
 BASIS, 1167
 GAUSS_ORDER, 1167
 NUMBER_OF_GAUSS_XI, 1168
 NUMBER_OF_SCHEMES, 1168
 QUADRATURE_SCHEME_MAP, 1168
 SCHEMES, 1168
 TYPE, 1168
 TYPES::REGION_PTR_TYPE, 1169
 PTR, 1169
 TYPES::REGION_TYPE, 1170
 COORDINATE_SYSTEM, 1171
 EQUATIONS_SETS, 1171
 FIELDS, 1171
 LABEL, 1171
 MESHES, 1171

NODES, 1171
NUMBER_OF_SUB_REGIONS, 1171
PARENT_REGION, 1171
REGION_FINISHED, 1171
SUB_REGIONS, 1172
USER_NUMBER, 1172
TYPES::SOLUTION_MAPPING_CREATE_-
VALUES_CACHE_TYPE, 1173
MATRIX_VARIABLE_TYPES, 1173
TYPES::SOLUTION_MAPPING_TYPE, 1174
CREATE_VALUES_CACHE, 1175
EQUATIONS_SET_TO_SOLVER_MAP,
1175
EQUATIONS_SETS, 1175
NUMBER_OF_EQUATIONS_SETS, 1175
NUMBER_OF_ROWS, 1175
NUMBER_OF_SOLVER_MATRICES, 1175
ROW_DOFS_MAPPING, 1175
SOLUTION, 1176
SOLUTION_MAPPING_FINISHED, 1176
SOLVER_COL_TO_EQUATIONS_SETS_-
MAP, 1176
SOLVER_ROW_TO_EQUATIONS_SET_-
MAPS, 1176
TOTAL_NUMBER_OF_ROWS, 1176
TYPES::SOLUTION_PTR_TYPE, 1177
PTR, 1177
TYPES::SOLUTION_TYPE, 1178
EQUATIONS_SET_ADDED_INDEX, 1178
EQUATIONS_SET_TO_ADD, 1178
PROBLEM, 1178
SOLUTION_FINISHED, 1179
SOLUTION_MAPPING, 1179
SOLUTION_NUMBER, 1179
SOLVER, 1179
TYPES::SOLVER_COL_TO_EQUATIONS_-
MAP_TYPE, 1180
COUPLING_COEFFICIENTS, 1180
EQUATIONS_COL_NUMBERS, 1180
EQUATIONS_MATRIX_NUMBERS, 1180
NUMBER_OF_EQUATIONS_MATRICES,
1181
TYPES::SOLVER_COL_TO_EQUATIONS_-
SET_MAP_TYPE, 1182
EQUATIONS, 1182
SOLVER_COL_TO_EQUATIONS_MAPS,
1182
TYPES::SOLVER_COL_TO_EQUATIONS_-
SETS_MAP_TYPE, 1183
COLUMN_DOFS_MAPPING, 1184
NUMBER_OF_COLUMNS, 1184
NUMBER_OF_DOFS, 1184
SOLUTION_MAPPING, 1184
SOLVER_COL_TO_EQUATIONS_SET_-
MAPS, 1184
SOLVER_COL_TO_VARIABLE_MAPS,
1184
SOLVER_MATRIX, 1184
SOLVER_MATRIX_NUMBER, 1184
TOTAL_NUMBER_OF_DOFs, 1185
TYPES::SOLVER_COL_TO_VARIABLE_MAP_-
TYPE, 1186
ADDITIVE_CONSTANT, 1186
EQUATIONS_SET_INDICES, 1186
NUMBER_OF_EQUATIONS_SETS, 1187
VARIABLE, 1187
VARIABLE_COEFFICIENT, 1187
VARIABLE_DOF, 1187
TYPES::SOLVER_MATRICES_TYPE, 1188
LIBRARY_TYPE, 1188
MATRICES, 1188
NUMBER_OF_MATRICES, 1189
NUMBER_OF_ROWS, 1189
RHS_VECTOR, 1189
SOLUTION_MAPPING, 1189
SOLVER, 1189
SOLVER_MATRICES_FINISHED, 1189
TOTAL_NUMBER_OF_ROWS, 1189
UPDATE_RHS_VECTOR, 1189
TYPES::SOLVER_MATRIX_PTR_TYPE, 1191
PTR, 1191
TYPES::SOLVER_MATRIX_TYPE, 1192
MATRIX, 1192
MATRIX_NUMBER, 1192
NUMBER_OF_COLUMNS, 1192
SOLVER_MATRICES, 1193
SOLVER_VECTOR, 1193
STORAGE_TYPE, 1193
UPDATE_MATRIX, 1193
TYPES::SOLVER_ROW_TO_EQUATIONS_-
SET_MAP_TYPE, 1194
COUPLING_COEFFICIENTS, 1194
EQUATIONS_ROW_NUMBER, 1194
EQUATIONS_SET, 1194
NUMBER_OF_ROWS, 1194
TYPES::SOLVER_TYPE, 1196
EIGENPROBLEM_SOLVER, 1197
LINEAR_SOLVER, 1197
NONLINEAR_SOLVER, 1197
OUTPUT_TYPE, 1197
SOLUTION, 1197
SOLUTION_MAPPING, 1197
SOLVE_TYPE, 1197
SOLVER_FINISHED, 1198
SOLVER_MATRICES, 1198
SPARSITY_TYPE, 1198
TIME_INTEGRATION_SOLVER, 1198

TYPES::SOURCE_EQUATIONS_MATRICES_-
 MAP_TYPE, 1199
 EQUATIONS_ROW_TO_SOURCE_DOF_-
 MAP, 1199
 SOURCE_DOF_TO_EQUATIONS_ROW_-
 MAP, 1199
 TYPES::TIME_INTEGRATION_SOLVER_-
 TYPE, 1200
 SOLVER, 1200
 SOLVER_LIBRARY, 1200
 TYPES::VARIABLE_TO_EQUATIONS_-
 COLUMN_MAP_TYPE, 1201
 COLUMN_DOF, 1201
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1202
 DOF_TO_COLUMNS_MAPS, 1202
 DOF_TO_ROWS_MAP, 1202
 EQUATIONS_MATRIX_NUMBERS, 1203
 NUMBER_OF_EQUATIONS_MATRICES,
 1203
 VARIABLE, 1203
 VARIABLE_INDEX, 1203
 VARIABLE_TYPE, 1203
 TYPES::VARIABLE_TO_SOLVER_COL_MAP_-
 TYPE, 1204
 ADDITIVE_CONSTANTS, 1204
 COLUMN_NUMBERS, 1204
 COUPLING_COEFFICIENTS, 1204
 TYPES::VECTOR_TYPE, 1206
 DATA_DP, 1206
 DATA_INTG, 1206
 DATA_L, 1207
 DATA_SP, 1207
 DATA_TYPE, 1207
 ID, 1207
 N, 1207
 SIZE, 1207
 VECTOR_FINISHED, 1207

 UNPACKCHARACTERS
 F90C::interface, 760
 UnPackCharacters
 f90c_c.c, 1272
 UPDATE_MATRIX
 TYPES::EQUATIONS_MATRIX_TYPE,
 1050
 TYPES::SOLVER_MATRIX_TYPE, 1193
 UPDATE_RHS_VECTOR
 TYPES::SOLVER_MATRICES_TYPE, 1189
 UPDATE_VECTOR
 TYPES::EQUATIONS_MATRICES_TYPE,
 1045
 USER_CPU
 TIMER, 665

timer_c.c, 1380
 USER_ELEMENT_NODES
 TYPES::MESH_ELEMENT_TYPE, 1135
 USER_NUMBER
 TYPES::BASIS_TYPE, 957
 TYPES::COORDINATE_SYSTEM_TYPE,
 960
 TYPES::DECOMPOSITION_ELEMENT_-
 TYPE, 962
 TYPES::DECOMPOSITION_TYPE, 972
 TYPES::DOMAIN_NODE_TYPE, 1019
 TYPES::EQUATIONS_SET_TYPE, 1067
 TYPES::FIELD_TYPE, 1111
 TYPES::GENERATED_MESH_TYPE, 1122
 TYPES::MESH_ELEMENT_TYPE, 1135
 TYPES::MESH_NODE_TYPE, 1139
 TYPES::MESH_TYPE, 1148
 TYPES::NODE_TYPE, 1150
 TYPES::PROBLEM_TYPE, 1161
 TYPES::REGION_TYPE, 1172

VALUE
 TREES::TREE_NODE_TYPE, 943
 VALUES
 TYPES::FIELD_INTERPOLATED_POINT_-
 TYPE, 1092
 var_str
 ISO_VARYING_STRING, 451
 ISO_VARYING_STRING::var_str, 845

VARIABLE
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048
 TYPES::SOLVER_COL_TO_VARIABLE_-
 MAP_TYPE, 1187
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1203

VARIABLE_COEFFICIENT
 TYPES::SOLVER_COL_TO_VARIABLE_-
 MAP_TYPE, 1187

VARIABLE_DOF
 TYPES::FIELD_DOF_TO_PARAM_MAP_-
 TYPE, 1085
 TYPES::SOLVER_COL_TO_VARIABLE_-
 MAP_TYPE, 1187

VARIABLE_INDEX
 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1203

VARIABLE_NUMBER
 TYPES::FIELD_VARIABLE_TYPE, 1118
 VARIABLE_TO_EQUATIONS_MATRICES_-
 MAPS
 TYPES::EQUATIONS_MAPPING_TYPE,
 1042

VARIABLE_TO_SOLVER_COL_MAPS

TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076

VARIABLE_TYPE
 TYPES::EQUATIONS_MATRIX_TO_-
 VARIABLE_MAP_TYPE, 1048

 TYPES::FIELD_VARIABLE_TYPE, 1118

 TYPES::VARIABLE_TO_EQUATIONS_-
 MATRICES_MAP_TYPE, 1203

VARIABLE_TYPE_MAP
 TYPES::FIELD_TYPE, 1111

VARIABLE_TYPES
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076

 TYPES::FIELD_CREATE_VALUES_-
 CACHE_TYPE, 1081

VARIABLES
 TYPES::EQUATIONS_TO_SOLVER_-
 MATRIX_MAPS_SM_TYPE, 1076

 TYPES::FIELD_TYPE, 1111

VecAssemblyBegin
 CMISS_PETSC::interface, 723

VecAssemblyEnd
 CMISS_PETSC::interface, 723

VecCreate
 CMISS_PETSC::interface, 723

VecCreateGhost
 CMISS_PETSC::interface, 724

VecCreateGhostWithArray
 CMISS_PETSC::interface, 724

VecCreateMPI
 CMISS_PETSC::interface, 724

VecCreateMPIWithArray
 CMISS_PETSC::interface, 724

VecCreateSeq
 CMISS_PETSC::interface, 724

VecCreateSeqWithArray
 CMISS_PETSC::interface, 724

VecDestroy
 CMISS_PETSC::interface, 724

VecDuplicate
 CMISS_PETSC::interface, 724

VecGetArray
 CMISS_PETSC::interface, 724

VecGetArrayF90
 CMISS_PETSC::interface, 724

VecGetLocalSize
 CMISS_PETSC::interface, 725

VecGetOwnershipRange
 CMISS_PETSC::interface, 725

VecGetSize
 CMISS_PETSC::interface, 725

VecGetValues
 CMISS_PETSC::interface, 725

VecGhostGetLocalForm
 CMISS_PETSC::interface, 725

 CMISS_PETSC::interface, 725

 VecGhostRestoreLocalForm
 CMISS_PETSC::interface, 725

 VecGhostUpdateBegin
 CMISS_PETSC::interface, 725

 VecGhostUpdateEnd
 CMISS_PETSC::interface, 725

 VecRestoreArray
 CMISS_PETSC::interface, 725

 VecRestoreArrayF90
 CMISS_PETSC::interface, 725

 VecSet
 CMISS_PETSC::interface, 726

 VecSetFromOptions
 CMISS_PETSC::interface, 726

 VecSetLocalToGlobalMapping
 CMISS_PETSC::interface, 726

 VecSetSizes
 CMISS_PETSC::interface, 726

 VecSetValues
 CMISS_PETSC::interface, 726

 VecSetValuesLocal
 CMISS_PETSC::interface, 726

VECTOR
 TYPES::DISTRIBUTED_VECTOR_-
 PETSC_TYPE, 986

 TYPES::ELEMENT_VECTOR_TYPE, 1030

 TYPES::EQUATIONS_MATRICES_TYPE,
 1045

VECTOR_ALL_VALUES_SET_DP
 MATRIX_VECTOR, 528

 MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET, 903

VECTOR_ALL_VALUES_SET_INTG
 MATRIX_VECTOR, 528

 MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET, 903

VECTOR_ALL_VALUES_SET_L
 MATRIX_VECTOR, 529

 MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET, 903

VECTOR_ALL_VALUES_SET_SP
 MATRIX_VECTOR, 529

 MATRIX_VECTOR::VECTOR_ALL_-
 VALUES_SET, 904

VECTOR_CREATE_FINISH
 MATRIX_VECTOR, 530

VECTOR_CREATE_START
 MATRIX_VECTOR, 530

VECTOR_DATA_GET_DP
 MATRIX_VECTOR, 530

 MATRIX_VECTOR::VECTOR_DATA_GET,
 905

VECTOR_DATA_GET_INTG

MATRIX_VECTOR, 531
 MATRIX_VECTOR::VECTOR_DATA_GET,
 905
 VECTOR_DATA_GET_L
 MATRIX_VECTOR, 531
 MATRIX_VECTOR::VECTOR_DATA_GET,
 905
 VECTOR_DATA_GET_SP
 MATRIX_VECTOR, 532
 MATRIX_VECTOR::VECTOR_DATA_GET,
 906
 VECTOR_DATA_TYPE_SET
 MATRIX_VECTOR, 532
 VECTOR_DESTROY
 MATRIX_VECTOR, 533
 VECTOR_DUPLICATE
 MATRIX_VECTOR, 533
 VECTOR_FINALISE
 MATRIX_VECTOR, 533
 VECTOR_FINISHED
 TYPES::DISTRIBUTED_VECTOR_TYPE,
 992
 TYPES::VECTOR_TYPE, 1207
 VECTOR_INITIALISE
 MATRIX_VECTOR, 534
 VECTOR_SIZE_SET
 MATRIX_VECTOR, 534
 VECTOR_VALUES_GET_DP
 MATRIX_VECTOR, 535
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 907
 VECTOR_VALUES_GET_DP1
 MATRIX_VECTOR, 535
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 907
 VECTOR_VALUES_GET_INTG
 MATRIX_VECTOR, 535
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 907
 VECTOR_VALUES_GET_INTG1
 MATRIX_VECTOR, 536
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 908
 VECTOR_VALUES_GET_L
 MATRIX_VECTOR, 536
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 908
 VECTOR_VALUES_GET_L1
 MATRIX_VECTOR, 537
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 908
 VECTOR_VALUES_GET_SP
 MATRIX_VECTOR, 537
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 909
 VECTOR_VALUES_GET_SP1
 MATRIX_VECTOR, 537
 MATRIX_VECTOR::VECTOR_VALUES_-
 GET, 909
 VECTOR_VALUES_SET_DP
 MATRIX_VECTOR, 538
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 910
 VECTOR_VALUES_SET_DP1
 MATRIX_VECTOR, 538
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 910
 VECTOR_VALUES_SET_INTG
 MATRIX_VECTOR, 539
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 910
 VECTOR_VALUES_SET_INTG1
 MATRIX_VECTOR, 539
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 911
 VECTOR_VALUES_SET_L
 MATRIX_VECTOR, 539
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 911
 VECTOR_VALUES_SET_L1
 MATRIX_VECTOR, 540
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 911
 VECTOR_VALUES_SET_SP
 MATRIX_VECTOR, 540
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 912
 VECTOR_VALUES_SET_SP1
 MATRIX_VECTOR, 541
 MATRIX_VECTOR::VECTOR_VALUES_-
 SET, 912
 VecView
 CMISS_PETSC::interface, 726
 verify_CH_VS
 ISO_VARYING_STRING, 451
 ISO_VARYING_STRING::verify, 847
 verify_VS_CH
 ISO_VARYING_STRING, 451
 ISO_VARYING_STRING::verify, 847
 verify_VS_VS
 ISO_VARYING_STRING, 451
 ISO_VARYING_STRING::verify, 847
 VOLUMES
 TYPES::FIELD_GEOMETRIC_-
 PARAMETERS_TYPE, 1087
 VSTRING_TO_LOWERCASE_C
 STRINGS, 663

STRINGS::VSTRING_TO_LOWERCASE,
939
VSTRING_TO_LOWERCASE_VS
 STRINGS, 663
 STRINGS::VSTRING_TO_LOWERCASE,
 939
VSTRING_TO_UPPERCASE_C
 STRINGS, 664
 STRINGS::VSTRING_TO_UPPERCASE,
 940
VSTRING_TO_UPPERCASE_VS
 STRINGS, 664
 STRINGS::VSTRING_TO_UPPERCASE,
 940

WR
 INPUT_OUTPUT, 372–392
WRITE_BINARY_FILE_CHARACTER
 BINARY_FILE, 184
 BINARY_FILE::WRITE_BINARY_FILE,
 709
WRITE_BINARY_FILE_DP
 BINARY_FILE, 184
 BINARY_FILE::WRITE_BINARY_FILE,
 709
WRITE_BINARY_FILE_DP1
 BINARY_FILE, 184
 BINARY_FILE::WRITE_BINARY_FILE,
 709
WRITE_BINARY_FILE_DPC
 BINARY_FILE, 184
 BINARY_FILE::WRITE_BINARY_FILE,
 709
WRITE_BINARY_FILE_DPC1
 BINARY_FILE, 185
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_INTG
 BINARY_FILE, 185
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_INTG1
 BINARY_FILE, 185
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_LINTG
 BINARY_FILE, 185
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_LINTG1
 BINARY_FILE, 185
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_LOGICAL
 BINARY_FILE, 186
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_LOGICAL1
 BINARY_FILE, 186
 BINARY_FILE::WRITE_BINARY_FILE,
 710
WRITE_BINARY_FILE_SINTG
 BINARY_FILE, 186
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_FILE_SINTG1
 BINARY_FILE, 186
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_FILE_SP
 BINARY_FILE, 187
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_FILE_SP1
 BINARY_FILE, 187
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_FILE_SPC
 BINARY_FILE, 187
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_FILE_SPC1
 BINARY_FILE, 187
 BINARY_FILE::WRITE_BINARY_FILE,
 711
WRITE_BINARY_TAG_HEADER
 BINARY_FILE, 187
WRITE_STR
 BASE_ROUTINES, 169
WRITE_STRING_C
 INPUT_OUTPUT, 393
 INPUT_OUTPUT::WRITE_STRING, 778
WRITE_STRING_FMT
 INPUT_OUTPUT, 393–397
WRITE_STRING_FMT_TWO_VALUE_C_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
WRITE_STRING_FMT_TWO_VALUE_C_DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
WRITE_STRING_FMT_TWO_VALUE_C_INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
WRITE_STRING_FMT_TWO_VALUE_C_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
WRITE_STRING_FMT_TWO_VALUE_C_SP

INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_C_VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_-
 INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_SP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_DP_VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_-
 DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_-
 INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_-
 SP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_INTG_-
 VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_SP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_L_VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_-
 INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_SP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_SP_VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_C
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_DP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_-
 INTG
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_L
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_SP
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_TWO_VALUE_VS_VS
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 TWO_VALUE, 781
 WRITE_STRING_FMT_VALUE_C
 INPUT_OUTPUT, 398
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 782
 WRITE_STRING_FMT_VALUE_DP

INPUT_OUTPUT, 398
INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 782
WRITE_STRING_FMT_VALUE_INTG
 INPUT_OUTPUT, 399
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 783
WRITE_STRING_FMT_VALUE_L
 INPUT_OUTPUT, 399
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 783
WRITE_STRING_FMT_VALUE_LINTG
 INPUT_OUTPUT, 400
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 784
WRITE_STRING_FMT_VALUE_SP
 INPUT_OUTPUT, 400
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 784
WRITE_STRING_FMT_VALUE_VS
 INPUT_OUTPUT, 401
 INPUT_OUTPUT::WRITE_STRING_FMT_-
 VALUE, 784
WRITE_STRING_IDX
 INPUT_OUTPUT, 401-405
WRITE_STRING_IDX_VECTOR_DP
 INPUT_OUTPUT::WRITE_STRING_IDX_-
 VECTOR, 786
WRITE_STRING_IDX_VECTOR_INTG
 INPUT_OUTPUT::WRITE_STRING_IDX_-
 VECTOR, 786
WRITE_STRING_IDX_VECTOR_L
 INPUT_OUTPUT::WRITE_STRING_IDX_-
 VECTOR, 786
WRITE_STRING_IDX_VECTOR_LINTG
 INPUT_OUTPUT::WRITE_STRING_IDX_-
 VECTOR, 786
WRITE_STRING_IDX_VECTOR_SP
 INPUT_OUTPUT::WRITE_STRING_IDX_-
 VECTOR, 786
WRITE_STRING_MATRIX_DP
 INPUT_OUTPUT, 405
 INPUT_OUTPUT::WRITE_STRING_-
 MATRIX, 787
WRITE_STRING_MATRIX_INTG
 INPUT_OUTPUT, 407
 INPUT_OUTPUT::WRITE_STRING_-
 MATRIX, 788
WRITE_STRING_MATRIX_L
 INPUT_OUTPUT, 408
 INPUT_OUTPUT::WRITE_STRING_-
 MATRIX, 789
WRITE_STRING_MATRIX_LINTG
 INPUT_OUTPUT, 409
INPUT_OUTPUT::WRITE_STRING_-
 MATRIX, 790
INPUT_OUTPUT::WRITE_STRING_-
 INDICES, 75
INPUT_OUTPUT_MatrixNameIndexFormat,
 75
WRITE_STRING_MATRIX_NAME_ONLY
 INPUT_OUTPUT_MatrixNameIndexFormat,
 75
WRITE_STRING_MATRIX_SP
 INPUT_OUTPUT, 410
 INPUT_OUTPUT::WRITE_STRING_-
 MATRIX, 790
WRITE_STRING_TWO_VALUE_C_C
 INPUT_OUTPUT, 411
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 793
WRITE_STRING_TWO_VALUE_C_DP
 INPUT_OUTPUT, 412
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 794
WRITE_STRING_TWO_VALUE_C_INTG
 INPUT_OUTPUT, 413
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 794
WRITE_STRING_TWO_VALUE_C_L
 INPUT_OUTPUT, 413
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 794
WRITE_STRING_TWO_VALUE_C_SP
 INPUT_OUTPUT, 414
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 795
WRITE_STRING_TWO_VALUE_C_VS
 INPUT_OUTPUT, 414
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 795
WRITE_STRING_TWO_VALUE_DP_C
 INPUT_OUTPUT, 415
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 796
WRITE_STRING_TWO_VALUE_DP_DP
 INPUT_OUTPUT, 416
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 796
WRITE_STRING_TWO_VALUE_DP_INTG
 INPUT_OUTPUT, 416
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 797
WRITE_STRING_TWO_VALUE_DP_L
 INPUT_OUTPUT, 417
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 797
WRITE_STRING_TWO_VALUE_DP_SP

INPUT_OUTPUT, 417
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 798
 WRITE_STRING_TWO_VALUE_DP_VS
 INPUT_OUTPUT, 418
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 798
 WRITE_STRING_TWO_VALUE_INTG_C
 INPUT_OUTPUT, 418
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 799
 WRITE_STRING_TWO_VALUE_INTG_DP
 INPUT_OUTPUT, 419
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 799
 WRITE_STRING_TWO_VALUE_INTG_INTG
 INPUT_OUTPUT, 420
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 800
 WRITE_STRING_TWO_VALUE_INTG_L
 INPUT_OUTPUT, 420
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 800
 WRITE_STRING_TWO_VALUE_INTG_SP
 INPUT_OUTPUT, 421
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 801
 WRITE_STRING_TWO_VALUE_INTG_VS
 INPUT_OUTPUT, 421
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 801
 WRITE_STRING_TWO_VALUE_L_C
 INPUT_OUTPUT, 422
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 802
 WRITE_STRING_TWO_VALUE_L_DP
 INPUT_OUTPUT, 423
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 802
 WRITE_STRING_TWO_VALUE_L_INTG
 INPUT_OUTPUT, 423
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 803
 WRITE_STRING_TWO_VALUE_L_L
 INPUT_OUTPUT, 424
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 803
 WRITE_STRING_TWO_VALUE_L_SP
 INPUT_OUTPUT, 424
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 804
 WRITE_STRING_TWO_VALUE_L_VS
 INPUT_OUTPUT, 425
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 804
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 804
 WRITE_STRING_TWO_VALUE_SP_C
 INPUT_OUTPUT, 425
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 805
 WRITE_STRING_TWO_VALUE_SP_DP
 INPUT_OUTPUT, 426
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 805
 WRITE_STRING_TWO_VALUE_SP_INTG
 INPUT_OUTPUT, 427
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 806
 WRITE_STRING_TWO_VALUE_SP_L
 INPUT_OUTPUT, 427
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 806
 WRITE_STRING_TWO_VALUE_SP_SP
 INPUT_OUTPUT, 428
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 807
 WRITE_STRING_TWO_VALUE_SP_VS
 INPUT_OUTPUT, 428
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 807
 WRITE_STRING_TWO_VALUE_VS_C
 INPUT_OUTPUT, 429
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 808
 WRITE_STRING_TWO_VALUE_VS_DP
 INPUT_OUTPUT, 430
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 808
 WRITE_STRING_TWO_VALUE_VS_INTG
 INPUT_OUTPUT, 430
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 809
 WRITE_STRING_TWO_VALUE_VS_L
 INPUT_OUTPUT, 431
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 809
 WRITE_STRING_TWO_VALUE_VS_SP
 INPUT_OUTPUT, 431
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 810
 WRITE_STRING_TWO_VALUE_VS_VS
 INPUT_OUTPUT, 432
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 810
 WRITE_STRING_VALUE_C
 INPUT_OUTPUT, 433
 INPUT_OUTPUT::WRITE_STRING_-
 TWO_VALUE, 812

WRITE_STRING_VALUE_DP
 INPUT_OUTPUT, [433](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [812](#)

WRITE_STRING_VALUE_INTG
 INPUT_OUTPUT, [434](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [813](#)

WRITE_STRING_VALUE_L
 INPUT_OUTPUT, [434](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [813](#)

WRITE_STRING_VALUE_LINTG
 INPUT_OUTPUT, [435](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [813](#)

WRITE_STRING_VALUE_SP
 INPUT_OUTPUT, [435](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [814](#)

WRITE_STRING_VALUE_VS
 INPUT_OUTPUT, [436](#)
 INPUT_OUTPUT::WRITE_STRING_-
 VALUE, [814](#)

WRITE_STRING_VECTOR_DP
 INPUT_OUTPUT::WRITE_STRING_-
 VECTOR, [815](#)

WRITE_STRING_VECTOR_INTG
 INPUT_OUTPUT::WRITE_STRING_-
 VECTOR, [815](#)

WRITE_STRING_VECTOR_L
 INPUT_OUTPUT::WRITE_STRING_-
 VECTOR, [815](#)

WRITE_STRING_VECTOR_LINTG
 INPUT_OUTPUT::WRITE_STRING_-
 VECTOR, [815](#)

WRITE_STRING_VECTOR_SP
 INPUT_OUTPUT::WRITE_STRING_-
 VECTOR, [815](#)

WRITE_STRING_VS
 INPUT_OUTPUT, [436](#)
 INPUT_OUTPUT::WRITE_STRING, [778](#)

XI_DIRECTION
 TYPES::DECOMPOSITION_LINE_TYPE,
 [966](#)

XI_DIRECTION1
 TYPES::DOMAIN_FACE_TYPE, [1003](#)

XI_DIRECTION2
 TYPES::DOMAIN_FACE_TYPE, [1003](#)