

MQCPack

Generated by Doxygen 1.8.16

1 Modules Index	1
1.1 Modules List	1
2 Data Type Index	3
2.1 Class Hierarchy	3
3 Data Type Index	5
3.1 Data Types List	5
4 File Index	7
4.1 File List	7
5 Module Documentation	9
5.1 mqc_algebra Module Reference	9
5.1.1 Function/Subroutine Documentation	17
5.1.1.1 bin_coeff()	17
5.1.1.2 factorial()	18
5.1.1.3 matrix_symm2sq_complex()	18
5.1.1.4 matrix_symm2sq_integer()	19
5.1.1.5 matrix_symm2sq_real()	19
5.1.1.6 mqc_allocate_matrix()	19
5.1.1.7 mqc_allocate_r4tensor()	19
5.1.1.8 mqc_allocate_scalar()	20
5.1.1.9 mqc_allocate_vector()	20
5.1.1.10 mqc_complexscalaradd()	21
5.1.1.11 mqc_complexscalardivide()	21
5.1.1.12 mqc_complexscalarmultiply()	22
5.1.1.13 mqc_complexscalarsubtract()	23
5.1.1.14 mqc_complexvectorproduct()	23
5.1.1.15 mqc_crossproduct()	24
5.1.1.16 mqc_deallocate_matrix()	24
5.1.1.17 mqc_deallocate_r4tensor()	24
5.1.1.18 mqc_deallocate_scalar()	24
5.1.1.19 mqc_deallocate_vector()	25
5.1.1.20 mqc_elementmatrixdivide()	25
5.1.1.21 mqc_elementmatrixproduct()	25
5.1.1.22 mqc_elementvectorproduct()	25
5.1.1.23 mqc_givens_matrix()	25
5.1.1.24 mqc_input_complex_scalar()	26
5.1.1.25 mqc_input_integer_scalar()	26
5.1.1.26 mqc_input_real_scalar()	27

5.1.1.27 <code>mqc_integertscalar()</code>	28
5.1.1.28 <code>mqc_integerlescalar()</code>	28
5.1.1.29 <code>mqc_integerscalaradd()</code>	29
5.1.1.30 <code>mqc_integerscalardivide()</code>	30
5.1.1.31 <code>mqc_integerscalarmultiply()</code>	30
5.1.1.32 <code>mqc_integerscalarsubtract()</code>	31
5.1.1.33 <code>mqc_integervectorproduct()</code>	32
5.1.1.34 <code>mqc_length_vector()</code>	32
5.1.1.35 <code>mqc_matrix_cast_complex()</code>	32
5.1.1.36 <code>mqc_matrix_cast_real()</code>	32
5.1.1.37 <code>mqc_matrix_columns()</code>	32
5.1.1.38 <code>mqc_matrix_conjugate_transpose()</code>	33
5.1.1.39 <code>mqc_matrix_copy_complex2int()</code>	33
5.1.1.40 <code>mqc_matrix_copy_complex2real()</code>	33
5.1.1.41 <code>mqc_matrix_copy_int2complex()</code>	33
5.1.1.42 <code>mqc_matrix_copy_int2real()</code>	33
5.1.1.43 <code>mqc_matrix_copy_real2complex()</code>	33
5.1.1.44 <code>mqc_matrix_copy_real2int()</code>	34
5.1.1.45 <code>mqc_matrix_determinant()</code>	34
5.1.1.46 <code>mqc_matrix_diag2full()</code>	34
5.1.1.47 <code>mqc_matrix_diag2symm()</code>	34
5.1.1.48 <code>mqc_matrix_diagmatrix_put_complex()</code>	34
5.1.1.49 <code>mqc_matrix_diagmatrix_put_integer()</code>	34
5.1.1.50 <code>mqc_matrix_diagmatrix_put_real()</code>	35
5.1.1.51 <code>mqc_matrix_diagmatrix_put_vector()</code>	35
5.1.1.52 <code>mqc_matrix_diagonalize()</code>	35
5.1.1.53 <code>mqc_matrix_full2diag()</code>	35
5.1.1.54 <code>mqc_matrix_full2symm()</code>	35
5.1.1.55 <code>mqc_matrix_generalized_eigensystem()</code>	35
5.1.1.56 <code>mqc_matrix_havecomplex()</code>	36
5.1.1.57 <code>mqc_matrix_havediagonal()</code>	36
5.1.1.58 <code>mqc_matrix_havefull()</code>	36
5.1.1.59 <code>mqc_matrix_haveinteger()</code>	36
5.1.1.60 <code>mqc_matrix_havereal()</code>	36
5.1.1.61 <code>mqc_matrix_havesymmetric()</code>	36
5.1.1.62 <code>mqc_matrix_identity()</code>	37
5.1.1.63 <code>mqc_matrix_initialize()</code>	37
5.1.1.64 <code>mqc_matrix_inverse()</code>	37
5.1.1.65 <code>mqc_matrix_isallocated()</code>	37

5.1.1.66 mqc_matrix_matrix_at()	37
5.1.1.67 mqc_matrix_matrix_contraction()	38
5.1.1.68 mqc_matrix_matrix_put()	38
5.1.1.69 mqc_matrix_norm()	39
5.1.1.70 mqc_matrix_rms_max()	39
5.1.1.71 mqc_matrix_rows()	39
5.1.1.72 mqc_matrix_scalar_at()	39
5.1.1.73 mqc_matrix_scalar_put()	39
5.1.1.74 mqc_matrix_set()	40
5.1.1.75 mqc_matrix_sqrt()	40
5.1.1.76 mqc_matrix_storagetype()	40
5.1.1.77 mqc_matrix_svd()	40
5.1.1.78 mqc_matrix_symm2diag()	40
5.1.1.79 mqc_matrix_symm2full()	41
5.1.1.80 mqc_matrix_symm2full_func()	41
5.1.1.81 mqc_matrix_symmetrize()	41
5.1.1.82 mqc_matrix_symmmatrix_put_complex()	41
5.1.1.83 mqc_matrix_symmmatrix_put_integer()	41
5.1.1.84 mqc_matrix_symmmatrix_put_real()	41
5.1.1.85 mqc_matrix_symmsymmr4tensor_put_complex()	42
5.1.1.86 mqc_matrix_symmsymmr4tensor_put_real()	42
5.1.1.87 mqc_matrix_test_diagonal()	42
5.1.1.88 mqc_matrix_test_symmetric()	42
5.1.1.89 mqc_matrix_trace()	42
5.1.1.90 mqc_matrix_transpose()	42
5.1.1.91 mqc_matrix_vector_at()	43
5.1.1.92 mqc_matrix_vector_put()	43
5.1.1.93 mqc_matrixmatrixdotproduct()	43
5.1.1.94 mqc_matrixmatrixproduct()	43
5.1.1.95 mqc_matrixmatrixsubtract()	43
5.1.1.96 mqc_matrixmatrixsum()	44
5.1.1.97 mqc_matrixscalarproduct()	44
5.1.1.98 mqc_matrixvectordotproduct()	44
5.1.1.99 mqc_outer()	44
5.1.1.100 mqc_output_complex_scalar()	44
5.1.1.101 mqc_output_integer_scalar()	45
5.1.1.102 mqc_output_mqcscalar_scalar()	46
5.1.1.103 mqc_output_real_scalar()	46
5.1.1.104 mqc_print_matrix_algebra1()	47

5.1.1.105 mqc_print_r4tensor_algebra1()	47
5.1.1.106 mqc_print_scalar_algebra1()	48
5.1.1.107 mqc_print_vector_algebra1()	49
5.1.1.108 mqc_r4tensor_at()	49
5.1.1.109 mqc_r4tensor_havecomplex()	49
5.1.1.110 mqc_r4tensor_haveinteger()	49
5.1.1.111 mqc_r4tensor_havereal()	49
5.1.1.112 mqc_r4tensor_initialize()	50
5.1.1.113 mqc_r4tensor_put()	50
5.1.1.114 mqc_realgtscalar()	50
5.1.1.115 mqc_reallescalar()	51
5.1.1.116 mqc_realltscalar()	52
5.1.1.117 mqc_realscalaradd()	52
5.1.1.118 mqc_realscalardivide()	53
5.1.1.119 mqc_realscalarmultiply()	54
5.1.1.120 mqc_realscalarsubtract()	54
5.1.1.121 mqc_realvectorproduct()	55
5.1.1.122 mqc_scalar_acos()	55
5.1.1.123 mqc_scalar_asin()	56
5.1.1.124 mqc_scalar_atan()	56
5.1.1.125 mqc_scalar_atan2()	57
5.1.1.126 mqc_scalar_cmplx()	58
5.1.1.127 mqc_scalar_complex_conjugate()	58
5.1.1.128 mqc_scalar_complex_imagpart()	59
5.1.1.129 mqc_scalar_complex_realpart()	60
5.1.1.130 mqc_scalar_cos()	60
5.1.1.131 mqc_scalar_get_abs_value()	61
5.1.1.132 mqc_scalar_get_intrinsic_complex()	61
5.1.1.133 mqc_scalar_get_intrinsic_integer()	62
5.1.1.134 mqc_scalar_get_intrinsic_real()	63
5.1.1.135 mqc_scalar_get_random_value()	63
5.1.1.136 mqc_scalar_havecomplex()	64
5.1.1.137 mqc_scalar_haveinteger()	64
5.1.1.138 mqc_scalar_havereal()	65
5.1.1.139 mqc_scalar_isallocated()	66
5.1.1.140 mqc_scalar_sin()	66
5.1.1.141 mqc_scalar_sqrt()	67
5.1.1.142 mqc_scalar_tan()	68
5.1.1.143 mqc_scalaradd()	68

5.1.1.144 mqc_scalarcomplexadd()	69
5.1.1.145 mqc_scalarcomplexdivide()	69
5.1.1.146 mqc_scalarcomplexexponent()	70
5.1.1.147 mqc_scalarcomplexmultiply()	71
5.1.1.148 mqc_scalarcomplexsubtract()	71
5.1.1.149 mqc_scalardivide()	72
5.1.1.150 mqc_scalareq()	73
5.1.1.151 mqc_scalarexponent()	73
5.1.1.152 mqc_scalarge()	74
5.1.1.153 mqc_scalargt()	75
5.1.1.154 mqc_scalargtinteger()	76
5.1.1.155 mqc_scalargtreal()	76
5.1.1.156 mqc_scalarintegeradd()	77
5.1.1.157 mqc_scalarintegerdivide()	78
5.1.1.158 mqc_scalarintegerexponent()	78
5.1.1.159 mqc_scalarintegermultiply()	79
5.1.1.160 mqc_scalarintegersubtract()	80
5.1.1.161 mqc_scalarle()	80
5.1.1.162 mqc_scalarleinteger()	81
5.1.1.163 mqc_scalarlereal()	82
5.1.1.164 mqc_scalarlt()	83
5.1.1.165 mqc_scalarltreal()	84
5.1.1.166 mqc_scalarmatrixproduct()	84
5.1.1.167 mqc_scalarmultiply()	85
5.1.1.168 mqc_scalarne()	85
5.1.1.169 mqc_scalarrealadd()	86
5.1.1.170 mqc_scalarrealddivide()	87
5.1.1.171 mqc_scalarrealexponent()	87
5.1.1.172 mqc_scalarrealmultiply()	88
5.1.1.173 mqc_scalarrealsubtract()	89
5.1.1.174 mqc_scalarsubtract()	89
5.1.1.175 mqc_scalarvectordifference()	90
5.1.1.176 mqc_scalarvectorproduct()	90
5.1.1.177 mqc_scalarvectorsum()	90
5.1.1.178 mqc_set_array2tensor()	91
5.1.1.179 mqc_set_array2vector_complex()	91
5.1.1.180 mqc_set_array2vector_integer()	91
5.1.1.181 mqc_set_array2vector_real()	91
5.1.1.182 mqc_set_complexarray2matrix()	91

5.1.1.183 mqc_set_integerarray2matrix()	91
5.1.1.184 mqc_set_matrix2complexarray()	92
5.1.1.185 mqc_set_matrix2integerarray()	92
5.1.1.186 mqc_set_matrix2matrix()	92
5.1.1.187 mqc_set_matrix2realarray()	92
5.1.1.188 mqc_set_realarray2matrix()	92
5.1.1.189 mqc_set_vector2complexarray()	92
5.1.1.190 mqc_set_vector2integerarray()	93
5.1.1.191 mqc_set_vector2realarray()	93
5.1.1.192 mqc_set_vector2vector()	93
5.1.1.193 mqc_vector2diagmatrix()	93
5.1.1.194 mqc_vector_abs()	93
5.1.1.195 mqc_vector_argsort()	93
5.1.1.196 mqc_vector_cast_complex()	94
5.1.1.197 mqc_vector_cast_real()	94
5.1.1.198 mqc_vector_cmplx()	94
5.1.1.199 mqc_vector_complex_imagpart()	94
5.1.1.200 mqc_vector_complex_realpart()	94
5.1.1.201 mqc_vector_conjugate_transpose()	94
5.1.1.202 mqc_vector_copy_complex2int()	95
5.1.1.203 mqc_vector_copy_complex2real()	95
5.1.1.204 mqc_vector_copy_int2complex()	95
5.1.1.205 mqc_vector_copy_int2real()	95
5.1.1.206 mqc_vector_copy_real2complex()	95
5.1.1.207 mqc_vector_copy_real2int()	95
5.1.1.208 mqc_vector_havecomplex()	96
5.1.1.209 mqc_vector_haveinteger()	96
5.1.1.210 mqc_vector_havereal()	96
5.1.1.211 mqc_vector_initialize()	96
5.1.1.212 mqc_vector_isallocated()	96
5.1.1.213 mqc_vector_iscolumn()	96
5.1.1.214 mqc_vector_maxloc()	97
5.1.1.215 mqc_vector_maxval()	97
5.1.1.216 mqc_vector_minloc()	97
5.1.1.217 mqc_vector_minval()	97
5.1.1.218 mqc_vector_norm()	97
5.1.1.219 mqc_vector_pop()	97
5.1.1.220 mqc_vector_power()	98
5.1.1.221 mqc_vector_push()	98

5.1.1.222 mqc_vector_scalar_at()	98
5.1.1.223 mqc_vector_scalar_increment()	98
5.1.1.224 mqc_vector_scalar_put()	98
5.1.1.225 mqc_vector_shift()	98
5.1.1.226 mqc_vector_sort()	99
5.1.1.227 mqc_vector_sqrt()	99
5.1.1.228 mqc_vector_transpose()	99
5.1.1.229 mqc_vector_unshift()	99
5.1.1.230 mqc_vector_vector_at()	99
5.1.1.231 mqc_vector_vector_put()	99
5.1.1.232 mqc_vectorcomplexdivide()	100
5.1.1.233 mqc_vectorcomplexproduct()	100
5.1.1.234 mqc_vectorintegerdivide()	100
5.1.1.235 mqc_vectorintegerproduct()	100
5.1.1.236 mqc_vectormatrixdotproduct()	100
5.1.1.237 mqc_vectorrealdivide()	100
5.1.1.238 mqc_vectorrealproduct()	101
5.1.1.239 mqc_vectorscalardivide()	101
5.1.1.240 mqc_vectorscalarproduct()	101
5.1.1.241 mqc_vectorvectordifference()	101
5.1.1.242 mqc_vectorvectordotproduct()	101
5.1.1.243 mqc_vectorvectorsum()	101
5.1.1.244 symindexhash()	102
5.2 mqc_est Module Reference	102
5.2.1 Function/Subroutine Documentation	104
5.2.1.1 gen_det_str()	104
5.2.1.2 get_one_gamma_matrix()	104
5.2.1.3 mqc_build_ci_hamiltonian()	104
5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()	105
5.2.1.5 mqc_eigenvalues_add_name()	105
5.2.1.6 mqc_eigenvalues_allocate()	105
5.2.1.7 mqc_eigenvalues_array_name()	105
5.2.1.8 mqc_eigenvalues_array_type()	105
5.2.1.9 mqc_eigenvalues_at()	105
5.2.1.10 mqc_eigenvalues_dimension()	106
5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()	106
5.2.1.12 mqc_eigenvalues_has_alpha()	106
5.2.1.13 mqc_eigenvalues_has_beta()	106
5.2.1.14 mqc_eigenvalues_integral_multiply()	106

5.2.1.15 mqc_eigenvalues_isallocated()	106
5.2.1.16 mqc_eigenvalues_output_array()	107
5.2.1.17 mqc_eigenvalues_output_block()	107
5.2.1.18 mqc_eri_integral_contraction()	107
5.2.1.19 mqc_integral_add_name()	107
5.2.1.20 mqc_integral_allocate()	107
5.2.1.21 mqc_integral_array_name()	108
5.2.1.22 mqc_integral_array_type()	108
5.2.1.23 mqc_integral_at()	108
5.2.1.24 mqc_integral_conjugate_transpose()	108
5.2.1.25 mqc_integral_delete_energy_list()	108
5.2.1.26 mqc_integral_difference()	108
5.2.1.27 mqc_integral_dimension()	109
5.2.1.28 mqc_integral_eigenvalues_multiply()	109
5.2.1.29 mqc_integral_get_energy_list()	109
5.2.1.30 mqc_integral_has_alpha()	109
5.2.1.31 mqc_integral_has_alphabeta()	109
5.2.1.32 mqc_integral_has_beta()	109
5.2.1.33 mqc_integral_has_betaalpha()	110
5.2.1.34 mqc_integral_identity()	110
5.2.1.35 mqc_integral_initialize()	110
5.2.1.36 mqc_integral_integral_multiply()	110
5.2.1.37 mqc_integral_isallocated()	110
5.2.1.38 mqc_integral_matrix_multiply()	111
5.2.1.39 mqc_integral_norm()	111
5.2.1.40 mqc_integral_output_array()	111
5.2.1.41 mqc_integral_output_block()	111
5.2.1.42 mqc_integral_output_orbitals()	111
5.2.1.43 mqc_integral_scalar_multiply()	112
5.2.1.44 mqc_integral_set_energy_list()	112
5.2.1.45 mqc_integral_sum()	112
5.2.1.46 mqc_integral_swap_orbitals()	112
5.2.1.47 mqc_integral_transpose()	112
5.2.1.48 mqc_matrix_integral_multiply()	113
5.2.1.49 mqc_matrix_spinblockghf()	113
5.2.1.50 mqc_matrix_undospinblockghf_eigenvalues()	113
5.2.1.51 mqc_matrix_undospinblockghf_integral()	113
5.2.1.52 mqc_print_eigenvalues()	113
5.2.1.53 mqc_print_integral()	114

5.2.1.54 mqc_print_twoeris()	114
5.2.1.55 mqc_print_wavefunction()	114
5.2.1.56 mqc_scalar_integral_multiply()	114
5.2.1.57 mqc_scf_eigenvalues_power()	114
5.2.1.58 mqc_scf_integral_contraction()	115
5.2.1.59 mqc_scf_integral_determinant()	115
5.2.1.60 mqc_scf_integral_diagonalize()	115
5.2.1.61 mqc_scf_integral_generalized_eigensystem()	115
5.2.1.62 mqc_scf_integral_inverse()	115
5.2.1.63 mqc_scf_integral_trace()	115
5.2.1.64 mqc_scf_transformation_matrix()	116
5.2.1.65 mqc_twoeris_allocate()	116
5.2.1.66 mqc_twoeris_at()	116
5.2.1.67 slater_condon()	116
5.2.1.68 twoeri_trans()	117
6 Data Type Documentation	119
6.1 mqc_algebra::abs Interface Reference	119
6.1.1 Member Function/Subroutine Documentation	119
6.1.1.1 mqc_scalar_get_abs_value()	119
6.1.1.2 mqc_vector_abs()	120
6.2 mqc_algebra::acos Interface Reference	120
6.2.1 Member Function/Subroutine Documentation	120
6.2.1.1 mqc_scalar_acos()	120
6.3 mqc_algebra::aimag Interface Reference	121
6.3.1 Member Function/Subroutine Documentation	121
6.3.1.1 mqc_scalar_complex_imagpart()	121
6.3.1.2 mqc_vector_complex_imagpart()	122
6.4 mqc_algebra::asin Interface Reference	122
6.4.1 Member Function/Subroutine Documentation	122
6.4.1.1 mqc_scalar_asin()	122
6.5 mqc_algebra::assignment(=) Interface Reference	123
6.5.1 Member Function/Subroutine Documentation	124
6.5.1.1 mqc_input_complex_scalar()	124
6.5.1.2 mqc_input_integer_scalar()	124
6.5.1.3 mqc_input_real_scalar()	125
6.5.1.4 mqc_output_complex_scalar()	126
6.5.1.5 mqc_output_integer_scalar()	126
6.5.1.6 mqc_output_mqcscalar_scalar()	127

6.5.1.7 mqc_output_real_scalar()	128
6.5.1.8 mqc_set_array2tensor()	128
6.5.1.9 mqc_set_array2vector_complex()	129
6.5.1.10 mqc_set_array2vector_integer()	129
6.5.1.11 mqc_set_array2vector_real()	129
6.5.1.12 mqc_set_complexarray2matrix()	129
6.5.1.13 mqc_set_integerarray2matrix()	129
6.5.1.14 mqc_set_matrix2complexarray()	129
6.5.1.15 mqc_set_matrix2integerarray()	130
6.5.1.16 mqc_set_matrix2matrix()	130
6.5.1.17 mqc_set_matrix2realarray()	130
6.5.1.18 mqc_set_realarray2matrix()	130
6.5.1.19 mqc_set_vector2complexarray()	130
6.5.1.20 mqc_set_vector2integerarray()	130
6.5.1.21 mqc_set_vector2realarray()	131
6.5.1.22 mqc_set_vector2vector()	131
6.6 mqc_est::assignment(=) Interface Reference	131
6.6.1 Member Function/Subroutine Documentation	131
6.6.1.1 mqc_eigenvalues_output_array()	131
6.6.1.2 mqc_integral_output_array()	131
6.7 mqc_algebra::atan Interface Reference	132
6.7.1 Member Function/Subroutine Documentation	132
6.7.1.1 mqc_scalar_atan()	132
6.8 mqc_algebra::atan2 Interface Reference	132
6.8.1 Member Function/Subroutine Documentation	133
6.8.1.1 mqc_scalar_atan2()	133
6.9 mqc_algebra::cmplx Interface Reference	133
6.9.1 Member Function/Subroutine Documentation	134
6.9.1.1 mqc_scalar_cmplx()	134
6.9.1.2 mqc_vector_cmplx()	134
6.10 mqc_algebra::conjg Interface Reference	135
6.10.1 Member Function/Subroutine Documentation	135
6.10.1.1 mqc_scalar_complex_conjugate()	135
6.11 mqc_algebra::contraction Interface Reference	135
6.11.1 Member Function/Subroutine Documentation	136
6.11.1.1 mqc_matrix_matrix_contraction()	136
6.12 mqc_est::contraction Interface Reference	136
6.12.1 Member Function/Subroutine Documentation	136
6.12.1.1 mqc_eri_integral_contraction()	136

6.12.1.2 mqc_scf_integral_contraction()	136
6.13 mqc_algebra::cos Interface Reference	137
6.13.1 Member Function/Subroutine Documentation	137
6.13.1.1 mqc_scalar_cos()	137
6.14 mqc_algebra::dagger Interface Reference	137
6.14.1 Member Function/Subroutine Documentation	138
6.14.1.1 mqc_matrix_conjugate_transpose()	138
6.14.1.2 mqc_vector_conjugate_transpose()	138
6.15 mqc_est::dagger Interface Reference	138
6.15.1 Member Function/Subroutine Documentation	138
6.15.1.1 mqc_integral_conjugate_transpose()	138
6.16 mqc_algebra::dot_product Interface Reference	139
6.16.1 Member Function/Subroutine Documentation	139
6.16.1.1 mqc_vectorvectordotproduct()	139
6.17 mqc_est::dot_product Interface Reference	139
6.17.1 Member Function/Subroutine Documentation	139
6.17.1.1 mqc_eigenvalue_eigenvalue_dotproduct()	139
6.18 mqc_algebra::matmul Interface Reference	140
6.18.1 Member Function/Subroutine Documentation	140
6.18.1.1 mqc_matrixmatrixdotproduct()	140
6.18.1.2 mqc_matrixvectordotproduct()	140
6.18.1.3 mqc_vectormatrixdotproduct()	140
6.19 mqc_est::matmul Interface Reference	141
6.19.1 Member Function/Subroutine Documentation	141
6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()	141
6.19.1.2 mqc_eigenvalues_integral_multiply()	141
6.19.1.3 mqc_integral_eigenvalues_multiply()	141
6.19.1.4 mqc_integral_integral_multiply()	142
6.19.1.5 mqc_integral_matrix_multiply()	142
6.19.1.6 mqc_matrix_integral_multiply()	142
6.20 mqc_algebra::matrix_symm2sq Interface Reference	142
6.20.1 Member Function/Subroutine Documentation	142
6.20.1.1 matrix_symm2sq_complex()	143
6.20.1.2 matrix_symm2sq_integer()	143
6.20.1.3 matrix_symm2sq_real()	143
6.21 mqc_algebra::mqc_cast_complex Interface Reference	143
6.21.1 Member Function/Subroutine Documentation	143
6.21.1.1 mqc_matrix_cast_complex()	143
6.21.1.2 mqc_vector_cast_complex()	144

6.22 mqc_algebra::mqc_cast_real Interface Reference	144
6.22.1 Member Function/Subroutine Documentation	144
6.22.1.1 mqc_matrix_cast_real()	144
6.22.1.2 mqc_vector_cast_real()	144
6.23 mqc_est::mqc_determinant Type Reference	144
6.23.1 Member Data Documentation	145
6.23.1.1 nalpstr	145
6.23.1.2 nbetstr	145
6.23.1.3 ndets	145
6.23.1.4 order	145
6.23.1.5 strings	145
6.24 mqc_est::mqc_determinant_string Type Reference	145
6.24.1 Member Data Documentation	146
6.24.1.1 alpha	146
6.24.1.2 beta	146
6.25 mqc_algebra::mqc_have_complex Interface Reference	146
6.25.1 Member Function/Subroutine Documentation	146
6.25.1.1 mqc_matrix_havecomplex()	146
6.25.1.2 mqc_vector_havecomplex()	146
6.26 mqc_algebra::mqc_have_int Interface Reference	147
6.26.1 Member Function/Subroutine Documentation	147
6.26.1.1 mqc_matrix_haveinteger()	147
6.26.1.2 mqc_vector_haveinteger()	147
6.27 mqc_algebra::mqc_have_real Interface Reference	147
6.27.1 Member Function/Subroutine Documentation	147
6.27.1.1 mqc_matrix_havereal()	148
6.27.1.2 mqc_vector_havereal()	148
6.28 mqc_algebra::mqc_matrix Type Reference	148
6.28.1 Member Function/Subroutine Documentation	149
6.28.1.1 at()	149
6.28.1.2 dagger()	149
6.28.1.3 det()	149
6.28.1.4 diag()	149
6.28.1.5 eigensys()	149
6.28.1.6 identity()	150
6.28.1.7 init()	150
6.28.1.8 initialize()	150
6.28.1.9 inv()	150
6.28.1.10 mat()	150

6.28.1.11 mput()	150
6.28.1.12 norm()	150
6.28.1.13 print()	151
6.28.1.14 put()	151
6.28.1.15 rmsmax()	151
6.28.1.16 s_type()	151
6.28.1.17 set()	151
6.28.1.18 sqrt()	151
6.28.1.19 svd()	151
6.28.1.20 trace()	152
6.28.1.21 transpose()	152
6.28.1.22 vat()	152
6.28.1.23 vput()	152
6.28.2 Member Data Documentation	152
6.28.2.1 matc	152
6.28.2.2 mati	152
6.28.2.3 matr	153
6.29 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference	153
6.29.1 Member Function/Subroutine Documentation	153
6.29.1.1 mqc_matrix_diagmatrix_put_complex()	153
6.29.1.2 mqc_matrix_diagmatrix_put_integer()	153
6.29.1.3 mqc_matrix_diagmatrix_put_real()	153
6.29.1.4 mqc_matrix_diagmatrix_put_vector()	154
6.30 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference	154
6.30.1 Member Function/Subroutine Documentation	154
6.30.1.1 mqc_matrix_symmmatrix_put_complex()	154
6.30.1.2 mqc_matrix_symmmatrix_put_integer()	154
6.30.1.3 mqc_matrix_symmmatrix_put_real()	154
6.31 mqc_est::mqc_matrix_undospinblockghf Interface Reference	155
6.31.1 Member Function/Subroutine Documentation	155
6.31.1.1 mqc_matrix_undospinblockghf_eigenvalues()	155
6.31.1.2 mqc_matrix_undospinblockghf_integral()	155
6.32 mqc_algebra::mqc_print Interface Reference	155
6.32.1 Member Function/Subroutine Documentation	155
6.32.1.1 mqc_print_matrix_algebra1()	156
6.32.1.2 mqc_print_r4tensor_algebra1()	156
6.32.1.3 mqc_print_scalar_algebra1()	156
6.32.1.4 mqc_print_vector_algebra1()	157
6.33 mqc_est::mqc_print Interface Reference	157

6.33.1 Member Function/Subroutine Documentation	158
6.33.1.1 <code>mqc_print_eigenvalues()</code>	158
6.33.1.2 <code>mqc_print_integral()</code>	158
6.33.1.3 <code>mqc_print_twoeris()</code>	158
6.33.1.4 <code>mqc_print_wavefunction()</code>	158
6.34 <code>mqc_est::mqc_pscf_wavefunction</code> Type Reference	159
6.34.1 Member Data Documentation	159
6.34.1.1 <code>nactive</code>	159
6.34.1.2 <code>ncore</code>	159
6.34.1.3 <code>nfrz</code>	159
6.34.1.4 <code>nval</code>	160
6.34.1.5 <code>pscf_amplitudes</code>	160
6.34.1.6 <code>pscf_energies</code>	160
6.35 <code>mqc_algebra::mqc_r4tensor</code> Type Reference	160
6.35.1 Member Function/Subroutine Documentation	160
6.35.1.1 <code>at()</code>	160
6.35.1.2 <code>init()</code>	161
6.35.1.3 <code>initialize()</code>	161
6.35.1.4 <code>print()</code>	161
6.35.1.5 <code>put()</code>	161
6.36 <code>mqc_algebra::mqc_scalar</code> Type Reference	161
6.36.1 Member Function/Subroutine Documentation	161
6.36.1.1 <code>abs()</code>	162
6.36.1.2 <code>cval()</code>	162
6.36.1.3 <code>ival()</code>	162
6.36.1.4 <code>print()</code>	162
6.36.1.5 <code>random()</code>	162
6.36.1.6 <code>rval()</code>	162
6.37 <code>mqc_est::mqc_scf_eigenvalues</code> Type Reference	163
6.37.1 Member Function/Subroutine Documentation	163
6.37.1.1 <code>addlabel()</code>	163
6.37.1.2 <code>at()</code>	163
6.37.1.3 <code>getblock()</code>	163
6.37.1.4 <code>getlabel()</code>	163
6.37.1.5 <code>power()</code>	163
6.37.1.6 <code>print()</code>	164
6.38 <code>mqc_est::mqc_scf_integral</code> Type Reference	164
6.38.1 Member Function/Subroutine Documentation	164
6.38.1.1 <code>addlabel()</code>	164

6.38.1.2 deleteelist()	164
6.38.1.3 det()	165
6.38.1.4 diag()	165
6.38.1.5 eigensys()	165
6.38.1.6 getblock()	165
6.38.1.7 getelist()	165
6.38.1.8 getlabel()	165
6.38.1.9 identity()	165
6.38.1.10 init()	166
6.38.1.11 inv()	166
6.38.1.12 norm()	166
6.38.1.13 orbitals()	166
6.38.1.14 print()	166
6.38.1.15 setelist()	166
6.38.1.16 swap()	166
6.38.1.17 trace()	167
6.39 mqc_algebra::mqc_set_array2vector Interface Reference	167
6.39.1 Member Function/Subroutine Documentation	167
6.39.1.1 mqc_set_array2vector_complex()	167
6.39.1.2 mqc_set_array2vector_integer()	167
6.39.1.3 mqc_set_array2vector_real()	167
6.40 mqc_est::mqc_twoeris Type Reference	168
6.40.1 Member Function/Subroutine Documentation	168
6.40.1.1 print()	168
6.41 mqc_algebra::mqc_vector Type Reference	168
6.41.1 Member Function/Subroutine Documentation	169
6.41.1.1 abs()	169
6.41.1.2 argsort()	169
6.41.1.3 at()	169
6.41.1.4 dagger()	169
6.41.1.5 diag()	169
6.41.1.6 init()	170
6.41.1.7 initialize()	170
6.41.1.8 maxloc()	170
6.41.1.9 maxval()	170
6.41.1.10 minloc()	170
6.41.1.11 minval()	170
6.41.1.12 norm()	170
6.41.1.13 pop()	171

6.41.1.14 power()	171
6.41.1.15 print()	171
6.41.1.16 push()	171
6.41.1.17 put()	171
6.41.1.18 shift()	171
6.41.1.19 size()	171
6.41.1.20 sort()	172
6.41.1.21 sqrt()	172
6.41.1.22 transpose()	172
6.41.1.23 unshift()	172
6.41.1.24 vat()	172
6.41.1.25 vput()	172
6.41.2 Member Data Documentation	172
6.41.2.1 data_type	172
6.41.2.2 length	173
6.41.2.3 vecc	173
6.41.2.4 veci	173
6.41.2.5 vecr	173
6.42 mqc_est::mqc_wavefunction Type Reference	173
6.42.1 Member Function/Subroutine Documentation	174
6.42.1.1 print()	174
6.42.2 Member Data Documentation	174
6.42.2.1 basis	174
6.42.2.2 charge	174
6.42.2.3 core_hamiltonian	175
6.42.2.4 density_matrix	175
6.42.2.5 fock_matrix	175
6.42.2.6 mo_coefficients	175
6.42.2.7 mo_energies	175
6.42.2.8 mo_symmetries	175
6.42.2.9 multiplicity	175
6.42.2.10 nalpha	176
6.42.2.11 nbasis	176
6.42.2.12 nbeta	176
6.42.2.13 nelectrons	176
6.42.2.14 overlap_matrix	176
6.42.2.15 scf_density_matrix	176
6.42.2.16 symmetry	176
6.42.2.17 wf_complex	177

6.42.2.18 wf_type	177
6.43 mqc_algebra::operator(*) Interface Reference	177
6.43.1 Member Function/Subroutine Documentation	178
6.43.1.1 mqc_complexscalarmultiply()	178
6.43.1.2 mqc_complexvectorproduct()	178
6.43.1.3 mqc_integerscalarmultiply()	179
6.43.1.4 mqc_integervectorproduct()	179
6.43.1.5 mqc_matrixmatrixproduct()	179
6.43.1.6 mqc_matrixscalarproduct()	180
6.43.1.7 mqc_realscalarmultiply()	180
6.43.1.8 mqc_realvectorproduct()	180
6.43.1.9 mqc_scalarcomplexmultiply()	181
6.43.1.10 mqc_scalarintegermultiply()	181
6.43.1.11 mqc_scalarmatrixproduct()	182
6.43.1.12 mqc_scalarmultiply()	182
6.43.1.13 mqc_scalarrealmultiply()	183
6.43.1.14 mqc_scalarvectorproduct()	183
6.43.1.15 mqc_vectorcomplexproduct()	184
6.43.1.16 mqc_vectorintegerproduct()	184
6.43.1.17 mqc_vectorrealproduct()	184
6.43.1.18 mqc_vectorscalarproduct()	184
6.44 mqc_est::operator(*) Interface Reference	184
6.44.1 Member Function/Subroutine Documentation	184
6.44.1.1 mqc_integral_scalar_multiply()	185
6.44.1.2 mqc_scalar_integral_multiply()	185
6.45 mqc_algebra::operator(**) Interface Reference	185
6.45.1 Member Function/Subroutine Documentation	185
6.45.1.1 mqc_scalarcomplexexponent()	185
6.45.1.2 mqc_scalarexponent()	186
6.45.1.3 mqc_scalarintegerexponent()	187
6.45.1.4 mqc_scalarrealexponent()	187
6.46 mqc_est::operator(+) Interface Reference	188
6.46.1 Member Function/Subroutine Documentation	188
6.46.1.1 mqc_integral_sum()	188
6.47 mqc_algebra::operator(+) Interface Reference	189
6.47.1 Member Function/Subroutine Documentation	189
6.47.1.1 mqc_complexscalaradd()	189
6.47.1.2 mqc_integerscalaradd()	190
6.47.1.3 mqc_matrixmatrixsum()	190

6.47.1.4	<code>mqc_realscalaradd()</code>	191
6.47.1.5	<code>mqc_scalaradd()</code>	191
6.47.1.6	<code>mqc_scalarcomplexadd()</code>	192
6.47.1.7	<code>mqc_scalarintegeradd()</code>	193
6.47.1.8	<code>mqc_scalarrealadd()</code>	193
6.47.1.9	<code>mqc_scalarvectorsum()</code>	194
6.47.1.10	<code>mqc_vectorvectorsum()</code>	194
6.48	<code>mqc_est::operator(-)</code> Interface Reference	194
6.48.1	Member Function/Subroutine Documentation	195
6.48.1.1	<code>mqc_integral_difference()</code>	195
6.49	<code>mqc_algebra::operator(-)</code> Interface Reference	195
6.49.1	Member Function/Subroutine Documentation	195
6.49.1.1	<code>mqc_complexscalarsubtract()</code>	195
6.49.1.2	<code>mqc_integerscalarsubtract()</code>	196
6.49.1.3	<code>mqc_matrixmatrixsubtract()</code>	197
6.49.1.4	<code>mqc_realscalarsubtract()</code>	197
6.49.1.5	<code>mqc_scalarcomplexsubtract()</code>	198
6.49.1.6	<code>mqc_scalarintegersubtract()</code>	198
6.49.1.7	<code>mqc_scalarrealsubtract()</code>	199
6.49.1.8	<code>mqc_scalarsubtract()</code>	200
6.49.1.9	<code>mqc_scalarvectordifference()</code>	200
6.49.1.10	<code>mqc_vectorvectordifference()</code>	201
6.50	<code>mqc_algebra::operator(.dot.)</code> Interface Reference	201
6.50.1	Member Function/Subroutine Documentation	201
6.50.1.1	<code>mqc_matrixmatrixdotproduct()</code>	201
6.50.1.2	<code>mqc_matrixvectordotproduct()</code>	201
6.50.1.3	<code>mqc_vectormatrixdotproduct()</code>	201
6.50.1.4	<code>mqc_vectorvectordotproduct()</code>	202
6.51	<code>mqc_algebra::operator(.eq.)</code> Interface Reference	202
6.51.1	Member Function/Subroutine Documentation	202
6.51.1.1	<code>mqc_scalareq()</code>	202
6.52	<code>mqc_algebra::operator(.ewd.)</code> Interface Reference	203
6.52.1	Member Function/Subroutine Documentation	203
6.52.1.1	<code>mqc_elementmatrixdivide()</code>	203
6.53	<code>mqc_algebra::operator(.ewp.)</code> Interface Reference	203
6.53.1	Member Function/Subroutine Documentation	203
6.53.1.1	<code>mqc_elementmatrixproduct()</code>	204
6.53.1.2	<code>mqc_elementvectorproduct()</code>	204
6.54	<code>mqc_algebra::operator(.ge.)</code> Interface Reference	204

6.54.1 Member Function/Subroutine Documentation	204
6.54.1.1 <code>mqc_scalarge()</code>	204
6.55 <code>mqc_algebra::operator(.gt.)</code> Interface Reference	205
6.55.1 Member Function/Subroutine Documentation	205
6.55.1.1 <code>mqc_integergtscalar()</code>	206
6.55.1.2 <code>mqc_realgtscalar()</code>	206
6.55.1.3 <code>mqc_scalargt()</code>	207
6.55.1.4 <code>mqc_scalargtinteger()</code>	208
6.55.1.5 <code>mqc_scalargtreal()</code>	209
6.56 <code>mqc_algebra::operator(.le.)</code> Interface Reference	210
6.56.1 Member Function/Subroutine Documentation	210
6.56.1.1 <code>mqc_integerlescalar()</code>	210
6.56.1.2 <code>mqc_reallescalar()</code>	211
6.56.1.3 <code>mqc_scalarle()</code>	212
6.56.1.4 <code>mqc_scalarleinteger()</code>	212
6.56.1.5 <code>mqc_scalarlereal()</code>	213
6.57 <code>mqc_algebra::operator(.lt.)</code> Interface Reference	214
6.57.1 Member Function/Subroutine Documentation	214
6.57.1.1 <code>mqc_realltscalar()</code>	214
6.57.1.2 <code>mqc_scalarlt()</code>	215
6.57.1.3 <code>mqc_scalarltreal()</code>	216
6.58 <code>mqc_algebra::operator(.ne.)</code> Interface Reference	217
6.58.1 Member Function/Subroutine Documentation	217
6.58.1.1 <code>mqc_scalarne()</code>	217
6.59 <code>mqc_algebra::operator(.outer.)</code> Interface Reference	218
6.59.1 Member Function/Subroutine Documentation	218
6.59.1.1 <code>mqc_outer()</code>	218
6.60 <code>mqc_algebra::operator(.x.)</code> Interface Reference	218
6.60.1 Member Function/Subroutine Documentation	218
6.60.1.1 <code>mqc_crossproduct()</code>	218
6.61 <code>mqc_algebra::operator(/)</code> Interface Reference	219
6.61.1 Member Function/Subroutine Documentation	219
6.61.1.1 <code>mqc_complexscalardivide()</code>	219
6.61.1.2 <code>mqc_integerscalardivide()</code>	220
6.61.1.3 <code>mqc_realscalardivide()</code>	220
6.61.1.4 <code>mqc_scalarcomplexdivide()</code>	221
6.61.1.5 <code>mqc_scalardivide()</code>	222
6.61.1.6 <code>mqc_scalarintegerdivide()</code>	222
6.61.1.7 <code>mqc_scalarrealdivide()</code>	223

6.61.1.8 mqc_vectorcomplexdivide()	224
6.61.1.9 mqc_vectorintegerdivide()	224
6.61.1.10 mqc_vectorrealddivide()	224
6.61.1.11 mqc_vectorscalardivide()	224
6.62 mqc_algebra::real Interface Reference	225
6.62.1 Member Function/Subroutine Documentation	225
6.62.1.1 mqc_scalar_complex_realpart()	225
6.62.1.2 mqc_vector_complex_realpart()	225
6.63 mqc_algebra::sin Interface Reference	226
6.63.1 Member Function/Subroutine Documentation	226
6.63.1.1 mqc_scalar_sin()	226
6.64 mqc_algebra::sqrt Interface Reference	226
6.64.1 Member Function/Subroutine Documentation	227
6.64.1.1 mqc_scalar_sqrt()	227
6.65 mqc_algebra::tan Interface Reference	227
6.65.1 Member Function/Subroutine Documentation	227
6.65.1.1 mqc_scalar_tan()	228
6.66 mqc_est::transpose Interface Reference	228
6.66.1 Member Function/Subroutine Documentation	228
6.66.1.1 mqc_integral_transpose()	229
6.67 mqc_algebra::transpose Interface Reference	229
6.67.1 Member Function/Subroutine Documentation	229
6.67.1.1 mqc_matrix_transpose()	229
6.67.1.2 mqc_vector_transpose()	229
7 File Documentation	231
7.1 src/mqc_algebra.F03 File Reference	231
7.2 src/mqc_est.F03 File Reference	239
Index	243

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

mqc_algebra	9
mqc_est	102

Chapter 2

Data Type Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

<code>mqc_algebra::abs</code>	119
<code>mqc_algebra::acos</code>	120
<code>mqc_algebra::aimag</code>	121
<code>mqc_algebra::asin</code>	122
<code>mqc_algebra::assignment(=)</code>	123
<code>mqc_est::assignment(=)</code>	131
<code>mqc_algebra::atan</code>	132
<code>mqc_algebra::atan2</code>	132
<code>mqc_algebra::cmplx</code>	133
<code>mqc_algebra::conjg</code>	135
<code>mqc_algebra::contraction</code>	135
<code>mqc_est::contraction</code>	136
<code>mqc_algebra::cos</code>	137
<code>mqc_algebra::dagger</code>	137
<code>mqc_est::dagger</code>	138
<code>mqc_algebra::dot_product</code>	139
<code>mqc_est::dot_product</code>	139
<code>mqc_algebra::matmul</code>	140
<code>mqc_est::matmul</code>	141
<code>mqc_algebra::matrix_symm2sq</code>	142
<code>mqc_algebra::mqc_cast_complex</code>	143
<code>mqc_algebra::mqc_cast_real</code>	144
<code>mqc_est::mqc_determinant</code>	144
<code>mqc_est::mqc_determinant_string</code>	145
<code>mqc_algebra::mqc_have_complex</code>	146
<code>mqc_algebra::mqc_have_int</code>	147
<code>mqc_algebra::mqc_have_real</code>	147
<code>mqc_algebra::mqc_matrix</code>	148
<code>mqc_algebra::mqc_matrix_diagmatrix_put</code>	153
<code>mqc_algebra::mqc_matrix_symmmatrix_put</code>	154
<code>mqc_est::mqc_matrix_undospinblockghf</code>	155

mqc_algebra::mqc_print	155
mqc_est::mqc_print	157
mqc_algebra::mqc_r4tensor	160
mqc_algebra::mqc_scalar	161
mqc_est::mqc_scf_eigenvalues	163
mqc_est::mqc_scf_integral	164
mqc_algebra::mqc_set_array2vector	167
mqc_est::mqc_twoeris	168
mqc_algebra::mqc_vector	168
mqc_est::mqc_wavefunction	173
mqc_est::mqc_pscf_wavefunction	159
mqc_algebra::operator(*)	177
mqc_est::operator(*)	184
mqc_algebra::operator(**)	185
mqc_est::operator(+)	188
mqc_algebra::operator(+)	189
mqc_est::operator(-)	194
mqc_algebra::operator(-)	195
mqc_algebra::operator(.dot.)	201
mqc_algebra::operator(.eq.)	202
mqc_algebra::operator(.ewd.)	203
mqc_algebra::operator(.ewp.)	203
mqc_algebra::operator(.ge.)	204
mqc_algebra::operator(.gt.)	205
mqc_algebra::operator(.le.)	210
mqc_algebra::operator(.lt.)	214
mqc_algebra::operator(.ne.)	217
mqc_algebra::operator(.outer.)	218
mqc_algebra::operator(.x.)	218
mqc_algebra::operator(/)	219
mqc_algebra::real	225
mqc_algebra::sin	226
mqc_algebra::sqrt	226
mqc_algebra::tan	227
mqc_est::transpose	228
mqc_algebra::transpose	229

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

mqc_algebra::abs	119
mqc_algebra::acos	120
mqc_algebra::aimag	121
mqc_algebra::asin	122
mqc_algebra::assignment(=)	123
mqc_est::assignment(=)	131
mqc_algebra::atan	132
mqc_algebra::atan2	132
mqc_algebra::cmplx	133
mqc_algebra::conjg	135
mqc_algebra::contraction	135
mqc_est::contraction	136
mqc_algebra::cos	137
mqc_algebra::dagger	137
mqc_est::dagger	138
mqc_algebra::dot_product	139
mqc_est::dot_product	139
mqc_algebra::matmul	140
mqc_est::matmul	141
mqc_algebra::matrix_symm2sq	142
mqc_algebra::mqc_cast_complex	143
mqc_algebra::mqc_cast_real	144
mqc_est::mqc_determinant	144
mqc_est::mqc_determinant_string	145
mqc_algebra::mqc_have_complex	146
mqc_algebra::mqc_have_int	147
mqc_algebra::mqc_have_real	147
mqc_algebra::mqc_matrix	148
mqc_algebra::mqc_matrix_diagmatrix_put	153
mqc_algebra::mqc_matrix_symmmatrix_put	154
mqc_est::mqc_matrix_undospinblockghf	155

mqc_algebra::mqc_print	155
mqc_est::mqc_print	157
mqc_est::mqc_pscf_wavefunction	159
mqc_algebra::mqc_r4tensor	160
mqc_algebra::mqc_scalar	161
mqc_est::mqc_scf_eigenvalues	163
mqc_est::mqc_scf_integral	164
mqc_algebra::mqc_set_array2vector	167
mqc_est::mqc_twoeris	168
mqc_algebra::mqc_vector	168
mqc_est::mqc_wavefunction	173
mqc_algebra::operator(*)	177
mqc_est::operator(*)	184
mqc_algebra::operator(**)	185
mqc_est::operator(+)	188
mqc_algebra::operator(+)	189
mqc_est::operator(-)	194
mqc_algebra::operator(-)	195
mqc_algebra::operator(.dot.)	201
mqc_algebra::operator(.eq.)	202
mqc_algebra::operator(.ewd.)	203
mqc_algebra::operator(.ewp.)	203
mqc_algebra::operator(.ge.)	204
mqc_algebra::operator(.gt.)	205
mqc_algebra::operator(.le.)	210
mqc_algebra::operator(.lt.)	214
mqc_algebra::operator(.ne.)	217
mqc_algebra::operator(.outer.)	218
mqc_algebra::operator(.x.)	218
mqc_algebra::operator(/)	219
mqc_algebra::real	225
mqc_algebra::sin	226
mqc_algebra::sqrt	226
mqc_algebra::tan	227
mqc_est::transpose	228
mqc_algebra::transpose	229

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/ mqc_algebra.F03	231
src/ mqc_est.F03	239

Chapter 5

Module Documentation

5.1 mqc_algebra Module Reference

Data Types

- interface [abs](#)
- interface [acos](#)
- interface [aimag](#)
- interface [asin](#)
- interface [assignment\(=\)](#)
- interface [atan](#)
- interface [atan2](#)
- interface [cmplx](#)
- interface [conjg](#)
- interface [contraction](#)
- interface [cos](#)
- interface [dagger](#)
- interface [dot_product](#)
- interface [matmul](#)
- interface [matrix_symm2sq](#)
- interface [mqc_cast_complex](#)
- interface [mqc_cast_real](#)
- interface [mqc_have_complex](#)
- interface [mqc_have_int](#)
- interface [mqc_have_real](#)
- type [mqc_matrix](#)
- interface [mqc_matrix_diagmatrix_put](#)
- interface [mqc_matrix_symmmatrix_put](#)
- interface [mqc_print](#)
- type [mqc_r4tensor](#)
- type [mqc_scalar](#)
- interface [mqc_set_array2vector](#)
- type [mqc_vector](#)
- interface [operator\(*\)](#)

- interface [operator\(**\)](#)
- interface [operator\(+\)](#)
- interface [operator\(-\)](#)
- interface [operator\(.dot.\)](#)
- interface [operator\(.eq.\)](#)
- interface [operator\(.ewd.\)](#)
- interface [operator\(.ewp.\)](#)
- interface [operator\(.ge.\)](#)
- interface [operator\(.gt.\)](#)
- interface [operator\(.le.\)](#)
- interface [operator\(.lt.\)](#)
- interface [operator\(.ne.\)](#)
- interface [operator\(.outer.\)](#)
- interface [operator\(.x.\)](#)
- interface [operator\(/\)](#)
- interface [real](#)
- interface [sin](#)
- interface [sqrt](#)
- interface [tan](#)
- interface [transpose](#)

Functions/Subroutines

- integer(kind=int64) function [factorial](#) (n)
Factorial returns the factorial of an integer
- integer(kind=int64) function [bin_coeff](#) (N, K)
Bin_Coeff returns the binomial coefficient of (n,k)
- subroutine [mqc_allocate_scalar](#) (Scalar, Data_type)
MQC_Allocate_Scalar is used to allocate a scalar type variable of the MQC_Scalar class
- subroutine [mqc_deallocate_scalar](#) (Scalar)
MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class
- logical function [mqc_scalar_isallocated](#) (Scalar)
MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar
- subroutine [mqc_input_integer_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an MQC_Scalar
- subroutine [mqc_input_real_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar
- subroutine [mqc_input_complex_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an MQC_Scalar
- subroutine [mqc_output_mqcscalar_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar
- subroutine [mqc_output_integer_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar
- subroutine [mqc_output_real_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar
- subroutine [mqc_output_complex_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar
- subroutine [mqc_print_scalar_algebra1](#) (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar

- type([mqc_scalar](#)) function [mqc_scalar_cmplx](#) (Scalar1, Scalar2)

MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars

- type([mqc_scalar](#)) function [mqc_scalar_sqrt](#) (Scalar)

MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_sin](#) (Scalar)

MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_cos](#) (Scalar)

MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_tan](#) (Scalar)

MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_asin](#) (Scalar)

MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_acos](#) (Scalar)

MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_atan](#) (Scalar)

MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar

- type([mqc_scalar](#)) function [mqc_scalar_atan2](#) (Scalar)

MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram

- logical function [mqc_scalar_havereal](#) (Scalar)

MQC_Scalar_HaveReal is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real

- logical function [mqc_scalar_haveinteger](#) (Scalar)

MQC_Scalar_HaveInteger is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer

- logical function [mqc_scalar_havecomplex](#) (Scalar)

MQC_Scalar_HaveComplex is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex

- [real](#)(kind=[real64](#)) function [mqc_scalar_get_intrinsic_real](#) (Scalar)

MQC_Scalar_Get_Intrinsic_Real is a function that returns the MQC_scalar value as an intrinsic real

- [integer](#)(kind=[int64](#)) function [mqc_scalar_get_intrinsic_integer](#) (Scalar)

MQC_Scalar_Get_Intrinsic_Integer is a function that returns the MQC_scalar value as an intrinsic integer

- [complex](#)(kind=[real64](#)) function [mqc_scalar_get_intrinsic_complex](#) (Scalar)

MQC_Scalar_Get_Intrinsic_Complex is a function that returns the MQC_scalar value as an intrinsic complex

- type([mqc_scalar](#)) function [mqc_scalar_get_abs_value](#) (Scalar)

MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable

- subroutine [mqc_scalar_get_random_value](#) (Scalar)

MQC_Scalar_Get_Random_Value is a function that returns a random real value from a uniform distribution between zero and one

- type([mqc_scalar](#)) function [mqc_scalaradd](#) (Scalar1, Scalar2)

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

- type([mqc_scalar](#)) function [mqc_scalarsubtract](#) (Scalar1, Scalar2)

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

- type([mqc_scalar](#)) function [mqc_scalarmultiply](#) (Scalar1, Scalar2)

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

- type([mqc_scalar](#)) function [mqc_scalardivide](#) (Scalar1, Scalar2)

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

- type([mqc_scalar](#)) function [mqc_scalarexponent](#) (Scalar1, Scalar2)
***MQC_ScalarExponent** is a function that raises one MQC_Scalar to the power of another MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarintegerexponent](#) (Scalar, IntIn)
***MQC_ScalarIntegerExponent** is a function that raises an MQC_Scalar to the power of an intrinsic integer*
- type([mqc_scalar](#)) function [mqc_scalarrealexponent](#) (Scalar, Realln)
***MQC_ScalarRealExponent** is a function that raises an MQC_Scalar to the power of an intrinsic real*
- type([mqc_scalar](#)) function [mqc_scalarcomplexexponent](#) (Scalar, Compln)
***MQC_ScalarComplexExponent** is a function that raises an MQC_Scalar to the power of an intrinsic complex*
- logical function [mqc_scalarne](#) (Scalar1, Scalar2)
***MQC_ScalarNE** is a function that returns TRUE if two MQC_Scalar variables are not equal*
- logical function [mqc_scalareq](#) (Scalar1, Scalar2)
***MQC_ScalarEQ** is a function that returns TRUE if two MQC_Scalar variables are equal*
- logical function [mqc_scalarlt](#) (Scalar1, Scalar2)
***MQC_ScalarLT** is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar*
- logical function [mqc_realltscalar](#) (Realln, Scalar)
***MQC_RealLTScalar** is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar*
- logical function [mqc_scalarltreal](#) (Scalar, Realln)
***MQC_ScalarLTReal** is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real*
- logical function [mqc_scalargt](#) (Scalar1, Scalar2)
***MQC_ScalarGT** is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar*
- logical function [mqc_integertgtscalar](#) (IntIn, Scalar)
***MQC_IntegerGTScalar** is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar*
- logical function [mqc_scalargtinteger](#) (Scalar, IntIn)
***MQC_ScalarGTInteger** is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer*
- logical function [mqc_realgtscalar](#) (Realln, Scalar)
***MQC_RealGTScalar** is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar*
- logical function [mqc_scalargtreal](#) (Scalar, Realln)
***MQC_ScalarGTReal** is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real*
- logical function [mqc_scalarle](#) (Scalar1, Scalar2)
***MQC_ScalarLE** is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar*
- logical function [mqc_reallescalar](#) (Realln, Scalar)
***MQC_RealLEScalar** is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar*
- logical function [mqc_scalarlereal](#) (Scalar, Realln)
***MQC_ScalarLEReal** is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real*
- logical function [mqc_integerlescalar](#) (IntIn, Scalar)
***MQC_IntegerLEScalar** is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar*
- logical function [mqc_scalarleinteger](#) (Scalar, IntIn)
***MQC_ScalarLEInteger** is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer*
- logical function [mqc_scalarge](#) (Scalar1, Scalar2)
***MQC_ScalarGE** is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalar_complex_conjugate](#) (ScalarIn)
***MQC_Scalar_Complex_Conjugate** is a function that returns the complex conjugate of an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalar_complex_realpart](#) (ScalarIn)
***MQC_Scalar_Complex_RealPart** is a function that returns the real part of an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalar_complex_imagpart](#) (ScalarIn)
***MQC_Scalar_Complex_ImagPart** is a function that returns the inaginary part of an MQC_Scalar*

- type([mqc_scalar](#)) function [mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
***MQC_IntegerScalarMultiply** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerMultiply** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_realscalarmultiply](#) (RealIn, Scalar)
***MQC_RealScalarMultiply** is a function that is used to multiply an intrinsic real by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarrealmultiply](#) (Scalar, RealIn)
***MQC_ScalarRealMultiply** is a function that is used to multiply an intrinsic real by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
***MQC_ComplexScalarMultiply** is a function that is used to multiply an intrinsic complex by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
***MQC_ScalarComplexMultiply** is a function that is used to multiply an intrinsic complex by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_integerscalardivide](#) (IntegerIn, Scalar)
***MQC_IntegerScalarDivide** is a function that is used to divide an intrinsic integer by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarintegerdivide](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerDivide** is a function that is used to divide an MQC_Scalar by an intrinsic integer*
- type([mqc_scalar](#)) function [mqc_realscalardivide](#) (RealIn, Scalar)
***MQC_RealScalarDivide** is a function that is used to divide an intrinsic real by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarrealdivide](#) (Scalar, RealIn)
***MQC_ScalarRealDivide** is a function that is used to divide an MQC_Scalar by an intrinsic real*
- type([mqc_scalar](#)) function [mqc_complexscalardivide](#) (ComplexIn, Scalar)
***MQC_ComplexScalarDivide** is a function that is used to divide an intrinsic complex by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarcomplexdivide](#) (Scalar, ComplexIn)
***MQC_ScalarComplexDivide** is a function that is used to divide an MQC_Scalar by an intrinsic complex*
- type([mqc_scalar](#)) function [mqc_integerscalaradd](#) (IntegerIn, Scalar)
***MQC_IntegerScalarAdd** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarintegeradd](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerAdd** is a function that is used to sum an intrinsic integer by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_realscalaradd](#) (RealIn, Scalar)
***MQC_RealScalarAdd** is a function that is used to sum an intrinsic real by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarrealadd](#) (Scalar, RealIn)
***MQC_ScalarRealAdd** is a function that is used to sum an intrinsic real by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_complexscalaradd](#) (ComplexIn, Scalar)
***MQC_ComplexScalarAdd** is a function that is used to sum an intrinsic complex by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarcomplexadd](#) (Scalar, ComplexIn)
***MQC_ScalarComplexAdd** is a function that is used to sum an intrinsic complex by an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_integerscalarsubtract](#) (IntegerIn, Scalar)
***MQC_IntegerScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic integer*
- type([mqc_scalar](#)) function [mqc_scalarintegersubtract](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerSubtract** is a function that is used to subtract an intrinsic integer from an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_realscalarsubtract](#) (RealIn, Scalar)
***MQC_RealScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic real*
- type([mqc_scalar](#)) function [mqc_scalarrealsubtract](#) (Scalar, RealIn)
***MQC_ScalarRealSubtract** is a function that is used to subtract an intrinsic real from an MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_complexscalarsubtract](#) (ComplexIn, Scalar)
***MQC_ComplexScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic complex*
- type([mqc_scalar](#)) function [mqc_scalarcomplexsubtract](#) (Scalar, ComplexIn)

MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar

- subroutine [mqc_allocate_vector](#) (N, Vector, Data_Type)
- subroutine [mqc_deallocate_vector](#) (Vector)
- integer(kind=int64) function [mqc_length_vector](#) (Vector)
- logical function [mqc_vector_havereal](#) (Vector)
- logical function [mqc_vector_haveinteger](#) (Vector)
- logical function [mqc_vector_havecomplex](#) (Vector)
- logical function [mqc_vector_iscolumn](#) (Vector)
- subroutine [mqc_vector_copy_int2real](#) (Vector)
- subroutine [mqc_vector_copy_int2complex](#) (Vector)
- subroutine [mqc_vector_copy_real2int](#) (Vector)
- subroutine [mqc_vector_copy_real2complex](#) (Vector)
- subroutine [mqc_vector_copy_complex2int](#) (Vector)
- subroutine [mqc_vector_copy_complex2real](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_scalar_at](#) (Vec, I)
- type([mqc_vector](#)) function [mqc_vector_vector_at](#) (Vec, I, J)
- subroutine [mqc_set_vector2integerarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_vector2realarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_vector2complexarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_vector2vector](#) (VectorOut, VectorIn)
- type([mqc_vector](#)) function [mqc_vectorvectorsum](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_vectorvectordifference](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_scalarvectorsum](#) (ScalarIn, VectorIn)
- type([mqc_vector](#)) function [mqc_scalarvectordifference](#) (ScalarIn, VectorIn)
- type([mqc_vector](#)) function [mqc_elementvectorproduct](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_vector_transpose](#) (Vector)
- type([mqc_vector](#)) function [mqc_vector_conjugate_transpose](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vectorvectordotproduct](#) (Vector1, Vector2)
- type([mqc_matrix](#)) function [mqc_outer](#) (VA, VB)
- type([mqc_vector](#)) function [mqc_crossproduct](#) (Vector1In, Vector2In)
- subroutine [mqc_print_vector_algebra1](#) (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- type([mqc_vector](#)) function [mqc_vector_cast_real](#) (VA)
- type([mqc_vector](#)) function [mqc_vector_cast_complex](#) (VA)
- subroutine [mqc_vector_scalar_put](#) (Vector, Scalar, I)
- subroutine [mqc_vector_scalar_increment](#) (Vector, Scalar, I)
- subroutine [mqc_vector_vector_put](#) (Vector, VectorIn, I)
- subroutine [mqc_vector_initialize](#) (Vector, Length, Scalar)
- type([mqc_vector](#)) function [mqc_scalarvectorproduct](#) (Scalar, Vector)
- type([mqc_vector](#)) function [mqc_vectorscalarproduct](#) (vector, scalar)
- type([mqc_vector](#)) function [mqc_vectorscalardivide](#) (vector, scalar)
- type([mqc_vector](#)) function [mqc_realvectorproduct](#) (Realln, Vector)
- type([mqc_vector](#)) function [mqc_vectorrealproduct](#) (vector, realln)
- type([mqc_vector](#)) function [mqc_vectorrealddivide](#) (vector, realln)
- type([mqc_vector](#)) function [mqc_integervectorproduct](#) (intln, Vector)
- type([mqc_vector](#)) function [mqc_vectorintegerproduct](#) (vector, intln)
- type([mqc_vector](#)) function [mqc_vectorintegerdivide](#) (vector, intln)
- type([mqc_vector](#)) function [mqc_complexvectorproduct](#) (Compln, Vector)
- type([mqc_vector](#)) function [mqc_vectorcomplexproduct](#) (vector, compln)

- type([mqc_vector](#)) function [mqc_vectorcomplexdivide](#) (vector, compln)
- type([mqc_scalar](#)) function [mqc_vector_norm](#) (vector, methodIn)
- logical function [mqc_vector_isallocated](#) (Vector)
- subroutine [mqc_vector_push](#) (Vector, Scalar)
- subroutine [mqc_vector_unshift](#) (Vector, Scalar)
- type([mqc_scalar](#)) function [mqc_vector_pop](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_shift](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_maxval](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_minval](#) (Vector)
- integer function [mqc_vector_maxloc](#) (Vector)
- integer function [mqc_vector_minloc](#) (Vector)
- type([mqc_vector](#)) function [mqc_vector_argsort](#) (Vector)
- subroutine [mqc_vector_sort](#) (Vector, idx)
- subroutine [mqc_vector_sqrt](#) (A)
- type([mqc_vector](#)) function [mqc_vector_abs](#) (A)
- subroutine [mqc_vector_power](#) (A, P)
- type([mqc_vector](#)) function [mqc_vector_complex_realpart](#) (A)
- type([mqc_vector](#)) function [mqc_vector_complex_imagpart](#) (A)
- type([mqc_vector](#)) function [mqc_vector_cmplx](#) (Vector1, Vector2)
- character(len=64) function [mqc_matrix_storage_type](#) (Matrix)
- subroutine [mqc_matrix_diagonalize](#) (A, EVals, EVecs)
- type([mqc_matrix](#)) function [mqc_matrix_cast_real](#) (MA)
- type([mqc_matrix](#)) function [mqc_matrix_cast_complex](#) (MA)
- type([mqc_scalar](#)) function [mqc_matrix_scalar_at](#) (Mat, I, J)
- type([mqc_vector](#)) function [mqc_matrix_vector_at](#) (Mat, Rows, Cols)
- recursive subroutine [mqc_matrix_vector_put](#) (Mat, VectorIn, Rows, Cols)
- type([mqc_matrix](#)) function [mqc_matrix_matrix_at](#) (Mat, Rows, Cols)

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

- subroutine [mqc_matrix_diagmatrix_put_vector](#) (diagVectorIn, mat)
- subroutine [mqc_matrix_diagmatrix_put_integer](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_real](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_complex](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_integer](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_real](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_complex](#) (mat, symmMatrixIn)
- recursive subroutine [mqc_matrix_matrix_put](#) (Mat, MatrixIn, Rows, Cols)
- integer(kind=int64) function [symindexhash](#) (i, j, k, l)
- type([mqc_matrix](#)) function [mqc_elementmatrixproduct](#) (A, B)
- type([mqc_matrix](#)) function [mqc_elementmatrixdivide](#) (A, B)
- logical function [mqc_matrix_test_symmetric](#) (Matrix, Option)
- logical function [mqc_matrix_test_diagonal](#) (Matrix)
- subroutine [mqc_allocate_matrix](#) (M, N, Matrix, Data_Type, Storage)
- subroutine [mqc_deallocate_matrix](#) (Matrix)
- logical function [mqc_matrix_isallocated](#) (Matrix)
- subroutine [mqc_set_integerarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_realarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_complexarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_matrix2integerarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2realarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2complexarray](#) (ArrayOut, MatrixIn)

- subroutine [mqc_set_matrix2matrix](#) (MatrixOut, MatrixIn)
- subroutine [mqc_print_matrix_algebra1](#) (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_matrix_copy_int2real](#) (Matrix)
- subroutine [mqc_matrix_copy_int2complex](#) (Matrix)
- subroutine [mqc_matrix_copy_real2int](#) (Matrix)
- subroutine [mqc_matrix_copy_real2complex](#) (Matrix)
- subroutine [mqc_matrix_copy_complex2int](#) (Matrix)
- subroutine [mqc_matrix_copy_complex2real](#) (Matrix)
- integer(kind=int64) function [mqc_matrix_rows](#) (Matrix)
- integer(kind=int64) function [mqc_matrix_columns](#) (Matrix)
- logical function [mqc_matrix_havereal](#) (Matrix)
- logical function [mqc_matrix_haveinteger](#) (Matrix)
- logical function [mqc_matrix_havecomplex](#) (Matrix)
- logical function [mqc_matrix_havfull](#) (Matrix)
- logical function [mqc_matrix_havesymmetric](#) (Matrix)
- logical function [mqc_matrix_havediagonal](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_transpose](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_conjugate_transpose](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_symmetrize](#) (Matrix)
- subroutine [mqc_matrix_full2symm](#) (Matrix)
- subroutine [mqc_matrix_symm2full](#) (Matrix, Option)
- subroutine [mqc_matrix_full2diag](#) (Matrix)
- subroutine [mqc_matrix_diag2full](#) (Matrix)
- subroutine [mqc_matrix_symm2diag](#) (Matrix)
- subroutine [mqc_matrix_diag2symm](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_symm2full_func](#) (Matrix)
- subroutine [matrix_symm2sq_integer](#) (N, I_Symm, I_Sq)
- subroutine [matrix_symm2sq_real](#) (N, A_Symm, A_Sq)
- subroutine [matrix_symm2sq_complex](#) (N, A_Symm, A_Sq)
- type([mqc_matrix](#)) function [mqc_vector2diagmatrix](#) (vector)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsum](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsubtract](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixproduct](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixdotproduct](#) (MA, MB)
- type([mqc_vector](#)) function [mqc_matrixvectordotproduct](#) (MA, VB)
- type([mqc_vector](#)) function [mqc_vectormatrixdotproduct](#) (VA, MB)
- type([mqc_matrix](#)) function [mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type([mqc_matrix](#)) function [mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type([mqc_scalar](#)) function [mqc_matrix_matrix_contraction](#) (Matrix1, Matrix2)
- subroutine [mqc_matrix_scalar_put](#) (Matrix, Scalar, I, J)
- subroutine [mqc_matrix_initialize](#) (Matrix, Rows, Columns, Scalar, Storage)
- subroutine [mqc_matrix_identity](#) (matrix, n, m)
- subroutine [mqc_matrix_set](#) (matrix, scalar, storage)
- type([mqc_scalar](#)) function [mqc_matrix_norm](#) (matrix, methodIn)
- type([mqc_scalar](#)) function [mqc_matrix_determinant](#) (a)
- type([mqc_matrix](#)) function [mqc_matrix_inverse](#) (a)
- type([mqc_scalar](#)) function [mqc_matrix_trace](#) (matrix)
- subroutine [mqc_matrix_generalized_eigensystem](#) (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine [mqc_matrix_svd](#) (A, EVals, EUVecs, EVVecs)
- subroutine [mqc_matrix_rms_max](#) (A, rms_A, max_A)
- subroutine [mqc_matrix_sqrt](#) (A, eVals, eVecs)

- type([mqc_matrix](#)) function [mqc_givens_matrix](#) (m_size, angle, p, q)
- subroutine [mqc_allocate_r4tensor](#) (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine [mqc_deallocate_r4tensor](#) (Tensor)
- type([mqc_scalar](#)) function [mqc_r4tensor_at](#) (Tensor, I, J, K, L)
- subroutine [mqc_r4tensor_put](#) (Tensor, Element, I, J, K, L)
- subroutine [mqc_print_r4tensor_algebra1](#) (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine [mqc_set_array2tensor](#) (TensorOut, ArrayIn)
- subroutine [mqc_r4tensor_initialize](#) (R4Tensor, I, J, K, L, Scalar)
- subroutine [mqc_matrix_symmsymmr4tensor_put_real](#) (r4Tensor, symmSymmMatrixIn)
- subroutine [mqc_matrix_symmsymmr4tensor_put_complex](#) (r4Tensor, symmSymmMatrixIn)
- logical function [mqc_r4tensor_haveinteger](#) (R4Tensor)
- logical function [mqc_r4tensor_havereal](#) (R4Tensor)
- logical function [mqc_r4tensor_havecomplex](#) (R4Tensor)

5.1.1 Function/Subroutine Documentation

5.1.1.1 bin_coeff()

```
integer(kind=int64) function mqc_algebra::bin_coeff (
    integer(kind=int64), intent(in) N,
    integer(kind=int64), intent(in) K )
```

Bin_Coeff returns the binomial coefficient of (n,k)

Purpose:

Bin_Coeff is a function that returns the binomial coefficient given input integer N and input integer K corresponding to N choose K.

Parameters

in	N	N is Integer(kind=int64) The number of objects
in	K	K is Integer(kind=int64) The number of permutations

Author

L. M. Thompson

Date

2016

5.1.1.2 factorial()

```
integer(kind=int64) function mqc_algebra::factorial (
    integer(kind=int64), intent(in) n )
```

Factorial returns the factorial of an integer

Purpose:

Factorial is a function that returns the factorial of an integer.

Parameters

in	<i>N</i>	N is Integer(kind=int64) The argument of the factorial function
----	----------	--

Author

L. M. Thompson

Date

2016

5.1.1.3 matrix_symm2sq_complex()

```
subroutine mqc_algebra::matrix_symm2sq_complex (
    integer(kind=int64), intent(in) N,
    complex(kind=real64), dimension(:), intent(in) A_Symm,
    complex(kind=real64), dimension(n,n), intent(out) A_Sq )
```


5.1.1.4 matrix_symm2sq_integer()

```
subroutine mqc_algebra::matrix_symm2sq_integer (
    integer(kind=int64), intent(in) N,
    integer(kind=int64), dimension(:), intent(in) I_Symm,
    integer(kind=int64), dimension(n,n), intent(out) I_Sq )
```

5.1.1.5 matrix_symm2sq_real()

```
subroutine mqc_algebra::matrix_symm2sq_real (
    integer(kind=int64), intent(in) N,
    real(kind=real64), dimension(:), intent(in) A_Symm,
    real(kind=real64), dimension(n,n), intent(out) A_Sq )
```

5.1.1.6 mqc_allocate_matrix()

```
subroutine mqc_algebra::mqc_allocate_matrix (
    integer(kind=int64), intent(in) M,
    integer(kind=int64), intent(in) N,
    class(mqc_matrix), intent(inout) Matrix,
    character(len=*), intent(in) Data_Type,
    character(len=*), intent(in) Storage )
```

5.1.1.7 mqc_allocate_r4tensor()

```
subroutine mqc_algebra::mqc_allocate_r4tensor (
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J,
    integer(kind=int64), intent(in) K,
    integer(kind=int64), intent(in) L,
    type(mqc_r4tensor), intent(inout) Tensor,
    character(len=*), intent(in) Data_Type,
    character(len=*), intent(in) Storage )
```

5.1.1.8 mqc_allocate_scalar()

```
subroutine mqc_algebra::mqc_allocate_scalar (
    type(mqc_scalar), intent(inout) Scalar,
    character(len=*), intent(in) Data_type )
```

MQC_Allocate_Scalar is used to allocate a scalar type variable of the **MQC_Scalar** class

Purpose:

MQC_Allocate_Scalar is a subroutine used to allocate a scalar type variable of the MQC_Scalar class. The following options are available:

1. Data_Type = 'Real' declares the MQC_Scalar variable to be of real type.
2. Data_Type = 'Integer' declares the MQC_Scalar variable to be of integer type.
3. Data_Type = 'Complex' declares the MQC_Scalar variable to be of complex type.

Parameters

in, out	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The name of the MQC_Scalar variable
in	<i>Data_Type</i>	Data_Type is Character(Len=*) = 'Real': the MQC_Scalar is real = 'Integer': the MQC_Scalar is integer = 'Complex': the MQC_Scalar is complex

Author

L. M. Thompson

Date

2016

5.1.1.9 mqc_allocate_vector()

```
subroutine mqc_algebra::mqc_allocate_vector (
    integer(kind=int64), intent(in) N,
    type(mqc_vector), intent(inout) Vector,
    character(len=*), intent(in) Data_Type )
```

5.1.1.10 mqc_complexscalaradd()

```
type(mqc_scalar) function mqc_algebra::mqc_complexscalaradd (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum

Author

L. M. Thompson

Date

2019

5.1.1.11 mqc_complexscalardivide()

```
type(mqc_scalar) function mqc_algebra::mqc_complexscalardivide (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>Complex</i> ↔ <i>In</i>	ComplexIn is Complex(kind=real64) The intrinsic complex variable numerator
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.12 mqc_complexscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_complexscalarmultiply (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>Complex</i> ↔ <i>In</i>	Complex is Complex(kind=real64) The intrinsic complex variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply

Author

L. M. Thompson

Date

2019

5.1.1.13 mqc_complexscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_complexscalarsubtract (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarSubtract is a function that is used to subtract an **MQC_Scalar** from an intrinsic complex

Purpose:

MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic complex.

Parameters

in	<i>ComplexIn</i>	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

5.1.1.14 mqc_complexvectorproduct()

```
type(mqc_vector) function mqc_algebra::mqc_complexvectorproduct (
    complex(kind=real64), intent(in) CompIn,
    type(mqc_vector), intent(in) Vector )
```

5.1.1.15 mqc_crossproduct()

```
type(mqc_vector) function mqc_algebra::mqc_crossproduct (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

5.1.1.16 mqc_deallocate_matrix()

```
subroutine mqc_algebra::mqc_deallocate_matrix (
    class(mqc_matrix), intent(inout) Matrix )
```

5.1.1.17 mqc_deallocate_r4tensor()

```
subroutine mqc_algebra::mqc_deallocate_r4tensor (
    type(mqc_r4tensor), intent(inout) Tensor )
```

5.1.1.18 mqc_deallocate_scalar()

```
subroutine mqc_algebra::mqc_deallocate_scalar (
    type(mqc_scalar), intent(inout) Scalar )
```

MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class

Purpose:

MQC_Deallocate_Scalar is a subroutine used to deallocate a scalar type variable of the MQC_Scalar class.

Parameters

in, out	Scalar	Scalar is Type(MQC_Scalar) The name of the MQC_Scalar variable to deallocate
---------	--------	---

Author

L. M. Thompson

Date

2016

5.1.1.19 mqc_deallocate_vector()

```
subroutine mqc_algebra::mqc_deallocate_vector (
    type(mqc_vector), intent(inout) Vector )
```

5.1.1.20 mqc_elementmatrixdivide()

```
type(mqc_matrix) function mqc_algebra::mqc_elementmatrixdivide (
    type(mqc_matrix), intent(in) A,
    type(mqc_matrix), intent(in) B )
```

5.1.1.21 mqc_elementmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::mqc_elementmatrixproduct (
    type(mqc_matrix), intent(in) A,
    type(mqc_matrix), intent(in) B )
```

5.1.1.22 mqc_elementvectorproduct()

```
type(mqc_vector) function mqc_algebra::mqc_elementvectorproduct (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

5.1.1.23 mqc_givens_matrix()

```
type(mqc_matrix) function mqc_algebra::mqc_givens_matrix (
    integer(kind=int64), intent(in) m_size,
    real(kind=real64), intent(in) angle,
    integer(kind=int64), intent(in) p,
    integer(kind=int64), intent(in) q )
```

5.1.1.24 `mqc_input_complex_scalar()`

```
subroutine mqc_algebra::mqc_input_complex_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    complex(kind=real64), intent(in) ScalarIn )
```

MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an **MQC_Scalar**

Purpose:

`MQC_Input_Complex_Scalar` is a subroutine is used to set an intrinsic complex to an `MQC_Scalar`.

Parameters

<code>in, out</code>	<i>ScalarOut</i>	<p><code>ScalarOut</code> is Type(<code>MQC_Scalar</code>) The name of the output variable</p>
<code>in</code>	<i>ScalarIn</i>	<p><code>ScalarIn</code> is Complex(kind=<code>real64</code>) The value of the input variable</p>

Author

L. M. Thompson

Date

2017

5.1.1.25 `mqc_input_integer_scalar()`

```
subroutine mqc_algebra::mqc_input_integer_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    integer(kind=int64), intent(in) ScalarIn )
```

MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an **MQC_Scalar**

Purpose:

`MQC_Input_Integer_Scalar` is a subroutine is used to set an intrinsic integer to an `MQC_Scalar`.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Integer(kind=int64) The value of the input variable

Author

L. M. Thompson

Date

2016

5.1.1.26 mqc_input_real_scalar()

```
subroutine mqc_algebra::mqc_input_real_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    real(kind=real64), intent(in) ScalarIn )
```

MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar

Purpose:

MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Real(kind=real64) The value of the input variable

Author

L. M. Thompson

Date

2016

5.1.1.27 mqc_integertgtscalar()

```
logical function mqc_algebra::mqc_integertgtscalar (
    integer(kind=int64), intent(in) IntIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar

Purpose:

MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer is greater than the real part of the MQC_Scalar and FALSE if the intrinsic integer is less than the real part of the MQC_Scalar. If the intrinsic integer is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise.

Parameters

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.28 mqc_integerlescalar()

```
logical function mqc_algebra::mqc_integerlescalar (
    integer(kind=int64), intent(in) IntIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar

Purpose:

MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic integer is greater than the real part of the MQC_Scalar.

Parameters

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.29 mqc_integerscalaradd()

```
type(mqc_scalar) function mqc_algebra::mqc_integerscalaradd (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

MQC_IntegerScalarAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

5.1.1.30 mqc_integerscalddivide()

```
type(mqc_scalar) function mqc_algebra::mqc_integerscalddivide (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar

Purpose:

MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable numerator
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.31 mqc_integerscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_integerscalarmultiply (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

`MQC_IntegerScalarMultiply` is a function that is used to multiply an intrinsic integer by an `MQC_Scalar`.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

5.1.1.32 mqc_integerscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_integerscalarsubtract (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarSubtract is a function that is used to subtract an `MQC_Scalar` from an intrinsic integer

Purpose:

`MQC_IntegerScalarSubtract` is a function that is used to subtract an `MQC_Scalar` from an intrinsic integer.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

5.1.1.33 mqc_integervectorproduct()

```
type(mqc_vector) function mqc_algebra::mqc_integervectorproduct (
    integer(kind=int64), intent(in) intIn,
    type(mqc_vector), intent(in) Vector )
```

5.1.1.34 mqc_length_vector()

```
integer(kind=int64) function mqc_algebra::mqc_length_vector (
    class(mqc_vector) Vector )
```

5.1.1.35 mqc_matrix_cast_complex()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_cast_complex (
    type(mqc_matrix), intent(in) MA )
```

5.1.1.36 mqc_matrix_cast_real()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_cast_real (
    type(mqc_matrix), intent(in) MA )
```

5.1.1.37 mqc_matrix_columns()

```
integer(kind=int64) function mqc_algebra::mqc_matrix_columns (
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.38 mqc_matrix_conjugate_transpose()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_conjugate_transpose (  
    class(mqc_matrix), intent(in) Matrix )
```

5.1.1.39 mqc_matrix_copy_complex2int()

```
subroutine mqc_algebra::mqc_matrix_copy_complex2int (  
    type(mqc_matrix) Matrix )
```

5.1.1.40 mqc_matrix_copy_complex2real()

```
subroutine mqc_algebra::mqc_matrix_copy_complex2real (  
    type(mqc_matrix) Matrix )
```

5.1.1.41 mqc_matrix_copy_int2complex()

```
subroutine mqc_algebra::mqc_matrix_copy_int2complex (  
    type(mqc_matrix) Matrix )
```

5.1.1.42 mqc_matrix_copy_int2real()

```
subroutine mqc_algebra::mqc_matrix_copy_int2real (  
    type(mqc_matrix) Matrix )
```

5.1.1.43 mqc_matrix_copy_real2complex()

```
subroutine mqc_algebra::mqc_matrix_copy_real2complex (  
    type(mqc_matrix) Matrix )
```

5.1.1.44 mqc_matrix_copy_real2int()

```
subroutine mqc_algebra::mqc_matrix_copy_real2int (  
    type(mqc_matrix) Matrix )
```

5.1.1.45 mqc_matrix_determinant()

```
type(mqc_scalar) function mqc_algebra::mqc_matrix_determinant (  
    class(mqc_matrix) a )
```

5.1.1.46 mqc_matrix_diag2full()

```
subroutine mqc_algebra::mqc_matrix_diag2full (  
    type(mqc_matrix), intent(inout) Matrix )
```

5.1.1.47 mqc_matrix_diag2symm()

```
subroutine mqc_algebra::mqc_matrix_diag2symm (  
    type(mqc_matrix), intent(inout) Matrix )
```

5.1.1.48 mqc_matrix_diagmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put_complex (  
    class(mqc_matrix), intent(inout) mat,  
    complex(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

5.1.1.49 mqc_matrix_diagmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put_integer (  
    class(mqc_matrix), intent(inout) mat,  
    integer(kind=int64), dimension(:), intent(in) diagMatrixIn )
```


5.1.1.50 mqc_matrix_diagmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put_real (
    class(mqc_matrix), intent(inout) mat,
    real(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

5.1.1.51 mqc_matrix_diagmatrix_put_vector()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put_vector (
    class(mqc_vector), intent(in) diagVectorIn,
    class(mqc_matrix), intent(inout) mat )
```

5.1.1.52 mqc_matrix_diagonalize()

```
subroutine mqc_algebra::mqc_matrix_diagonalize (
    class(mqc_matrix), intent(in) A,
    type(mqc_vector), intent(inout), optional EVals,
    type(mqc_matrix), intent(inout), optional EVecs )
```

5.1.1.53 mqc_matrix_full2diag()

```
subroutine mqc_algebra::mqc_matrix_full2diag (
    type(mqc_matrix), intent(inout) Matrix )
```

5.1.1.54 mqc_matrix_full2symm()

```
subroutine mqc_algebra::mqc_matrix_full2symm (
    type(mqc_matrix), intent(inout) Matrix )
```

5.1.1.55 mqc_matrix_generalized_eigensystem()

```
subroutine mqc_algebra::mqc_matrix_generalized_eigensystem (
    class(mqc_matrix), intent(inout) a,
    type(mqc_matrix), intent(inout), optional bIn,
    type(mqc_vector), intent(out), optional eigenvals,
    type(mqc_matrix), intent(out), optional reigenvecs,
    type(mqc_matrix), intent(out), optional leigenvecs )
```

5.1.1.56 mqc_matrix_havecomplex()

```
logical function mqc_algebra::mqc_matrix_havecomplex (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.57 mqc_matrix_havediagonal()

```
logical function mqc_algebra::mqc_matrix_havediagonal (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.58 mqc_matrix_havefull()

```
logical function mqc_algebra::mqc_matrix_havefull (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.59 mqc_matrix_haveinteger()

```
logical function mqc_algebra::mqc_matrix_haveinteger (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.60 mqc_matrix_havereal()

```
logical function mqc_algebra::mqc_matrix_havereal (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.61 mqc_matrix_havesymmetric()

```
logical function mqc_algebra::mqc_matrix_havesymmetric (  
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.62 mqc_matrix_identity()

```
subroutine mqc_algebra::mqc_matrix_identity (
    class(mqc_matrix), intent(inout) matrix,
    integer(kind=int64) n,
    integer(kind=int64) m )
```

5.1.1.63 mqc_matrix_initialize()

```
subroutine mqc_algebra::mqc_matrix_initialize (
    class(mqc_matrix), intent(inout) Matrix,
    integer(kind=int64), intent(in) Rows,
    integer(kind=int64), intent(in) Columns,
    class(*), optional Scalar,
    character(len=*), intent(in), optional Storage )
```

5.1.1.64 mqc_matrix_inverse()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_inverse (
    class(mqc_matrix) a )
```

5.1.1.65 mqc_matrix_isallocated()

```
logical function mqc_algebra::mqc_matrix_isallocated (
    class(mqc_matrix), intent(inout) Matrix )
```

5.1.1.66 mqc_matrix_matrix_at()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_matrix_at (
    class(mqc_matrix), intent(in) Mat,
    integer(kind=int64), dimension(:), intent(in) Rows,
    integer(kind=int64), dimension(:), intent(in) Cols )
```

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

Parameters

in	<i>Mat</i>	Mat is Class(MQC_Matrix) Name of the input matrix variable
in	<i>rows</i>	Rows is Integer(kind=int64),Dimension(:) If = [A,B]: output is submatrix of rows A to B If (A,B)>0 row count is from first index If (A,B)<0 row count is from last index If = [0]: submatrix of rows equivalent to [1,-1]
in	<i>Cols</i>	Cols is Integer(kind=int64),Dimension(:) If = [A,B]: output is submatrix of columns A to B If (A,B)>0 column count is from first index If (A,B)<0 column count is from last index If = [0]: submatrix of columns equivalent to [1,-1]

Author

L. M. Thompson

Date

2017

5.1.1.67 mqc_matrix_matrix_contraction()

```
type(mqc_scalar) function mqc_algebra::mqc_matrix_matrix_contraction (
    type(mqc_matrix), intent(in) Matrix1,
    type(mqc_matrix), intent(in) Matrix2 )
```

5.1.1.68 mqc_matrix_matrix_put()

```
recursive subroutine mqc_algebra::mqc_matrix_matrix_put (
    class(mqc_matrix), intent(inout) Mat,
    type(mqc_matrix), intent(in) MatrixIn,
    integer(kind=int64), dimension(:), intent(in) Rows,
    integer(kind=int64), dimension(:), intent(in) Cols )
```

5.1.1.69 mqc_matrix_norm()

```
type(mqc_scalar) function mqc_algebra::mqc_matrix_norm (
    class(mqc_matrix), intent(inout) matrix,
    character(len=1), intent(in), optional methodIn )
```

5.1.1.70 mqc_matrix_rms_max()

```
subroutine mqc_algebra::mqc_matrix_rms_max (
    class(mqc_matrix), intent(inout) A,
    type(mqc_scalar), intent(out) rms_A,
    type(mqc_scalar), intent(out) max_A )
```

5.1.1.71 mqc_matrix_rows()

```
integer(kind=int64) function mqc_algebra::mqc_matrix_rows (
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.72 mqc_matrix_scalar_at()

```
type(mqc_scalar) function mqc_algebra::mqc_matrix_scalar_at (
    class(mqc_matrix), intent(in) Mat,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J )
```

5.1.1.73 mqc_matrix_scalar_put()

```
subroutine mqc_algebra::mqc_matrix_scalar_put (
    class(mqc_matrix), intent(inout) Matrix,
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J )
```

5.1.1.74 mqc_matrix_set()

```
subroutine mqc_algebra::mqc_matrix_set (  
    class(mqc_matrix), intent(inout) matrix,  
    class(*), optional scalar,  
    character(len=*), intent(in), optional storage )
```

5.1.1.75 mqc_matrix_sqrt()

```
subroutine mqc_algebra::mqc_matrix_sqrt (  
    class(mqc_matrix), intent(inout) A,  
    type(mqc_vector), intent(inout), optional eVals,  
    type(mqc_matrix), intent(inout), optional eVecs )
```

5.1.1.76 mqc_matrix_storage_type()

```
character(len=64) function mqc_algebra::mqc_matrix_storage_type (  
    class(mqc_matrix), intent(in) Matrix )
```

5.1.1.77 mqc_matrix_svd()

```
subroutine mqc_algebra::mqc_matrix_svd (  
    class(mqc_matrix), intent(inout) A,  
    type(mqc_vector), intent(inout), optional EVals,  
    type(mqc_matrix), intent(inout), optional EUVecs,  
    type(mqc_matrix), intent(inout), optional EVVecs )
```

5.1.1.78 mqc_matrix_symm2diag()

```
subroutine mqc_algebra::mqc_matrix_symm2diag (  
    type(mqc_matrix), intent(inout) Matrix )
```

5.1.1.79 mqc_matrix_symm2full()

```
subroutine mqc_algebra::mqc_matrix_symm2full (
    type(mqc_matrix), intent(inout) Matrix,
    character(len=*), intent(in), optional Option )
```

5.1.1.80 mqc_matrix_symm2full_func()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_symm2full_func (
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.81 mqc_matrix_symmetrize()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_symmetrize (
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.82 mqc_matrix_symmmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put_complex (
    class(mqc_matrix), intent(inout) mat,
    complex(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

5.1.1.83 mqc_matrix_symmmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put_integer (
    class(mqc_matrix), intent(inout) mat,
    integer(kind=int64), dimension(:), intent(in) symmMatrixIn )
```

5.1.1.84 mqc_matrix_symmmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put_real (
    class(mqc_matrix), intent(inout) mat,
    real(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

5.1.1.85 mqc_matrix_symmsymmr4tensor_put_complex()

```
subroutine mqc_algebra::mqc_matrix_symmsymmr4tensor_put_complex (
    class(mqc_r4tensor), intent(inout) r4Tensor,
    complex(kind=real64), dimension(:), intent(in) symmSymmMatrixIn )
```

5.1.1.86 mqc_matrix_symmsymmr4tensor_put_real()

```
subroutine mqc_algebra::mqc_matrix_symmsymmr4tensor_put_real (
    class(mqc_r4tensor), intent(inout) r4Tensor,
    real(kind=real64), dimension(:), intent(in) symmSymmMatrixIn )
```

5.1.1.87 mqc_matrix_test_diagonal()

```
logical function mqc_algebra::mqc_matrix_test_diagonal (
    class(mqc_matrix), intent(in) Matrix )
```

5.1.1.88 mqc_matrix_test_symmetric()

```
logical function mqc_algebra::mqc_matrix_test_symmetric (
    class(mqc_matrix), intent(in) Matrix,
    character(len=*), intent(in), optional Option )
```

5.1.1.89 mqc_matrix_trace()

```
type(mqc_scalar) function mqc_algebra::mqc_matrix_trace (
    class(mqc_matrix), intent(in) matrix )
```

5.1.1.90 mqc_matrix_transpose()

```
type(mqc_matrix) function mqc_algebra::mqc_matrix_transpose (
    class(mqc_matrix), intent(in) Matrix )
```


5.1.1.91 mqc_matrix_vector_at()

```
type(mqc_vector) function mqc_algebra::mqc_matrix_vector_at (
    class(mqc_matrix), intent(in) Mat,
    integer(kind=int64), dimension(:), intent(in) Rows,
    integer(kind=int64), dimension(:), intent(in) Cols )
```

5.1.1.92 mqc_matrix_vector_put()

```
recursive subroutine mqc_algebra::mqc_matrix_vector_put (
    class(mqc_matrix), intent(inout) Mat,
    type(mqc_vector), intent(in) VectorIn,
    integer(kind=int64), dimension(:), intent(in) Rows,
    integer(kind=int64), dimension(:), intent(in) Cols )
```

5.1.1.93 mqc_matrixmatrixdotproduct()

```
type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixdotproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

5.1.1.94 mqc_matrixmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

5.1.1.95 mqc_matrixmatrixsubtract()

```
type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixsubtract (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

5.1.1.96 mqc_matrixmatrixsum()

```
type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixsum (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

5.1.1.97 mqc_matrixscalarproduct()

```
type(mqc_matrix) function mqc_algebra::mqc_matrixscalarproduct (
    type(mqc_matrix), intent(in) Matrix,
    type(mqc_scalar), intent(in) Scalar )
```

5.1.1.98 mqc_matrixvectordotproduct()

```
type(mqc_vector) function mqc_algebra::mqc_matrixvectordotproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_vector), intent(in) VB )
```

5.1.1.99 mqc_outer()

```
type(mqc_matrix) function mqc_algebra::mqc_outer (
    type(mqc_vector), intent(in) VA,
    type(mqc_vector), intent(in) VB )
```

5.1.1.100 mqc_output_complex_scalar()

```
subroutine mqc_algebra::mqc_output_complex_scalar (
    complex(kind=real64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar

Purpose:

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Complex(kind=real64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2017

5.1.1.101 mqc_output_integer_scalar()

```
subroutine mqc_algebra::mqc_output_integer_scalar (
    integer(kind=int64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar

Purpose:

MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Integer(kind=int64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

5.1.1.102 mqc_output_mqcscalar_scalar()

```
subroutine mqc_algebra::mqc_output_mqcscalar_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar

Purpose:

MQC_Output_MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

5.1.1.103 mqc_output_real_scalar()

```
subroutine mqc_algebra::mqc_output_real_scalar (
    real(kind=real64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar

Purpose:

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Real(kind=real64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

5.1.1.104 mqc_print_matrix_algebra1()

```
subroutine mqc_algebra::mqc_print_matrix_algebra1 (
    class(mqc_matrix), intent(in) Matrix,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )
```

5.1.1.105 mqc_print_r4tensor_algebra1()

```
subroutine mqc_algebra::mqc_print_r4tensor_algebra1 (
    class(mqc_r4tensor), intent(in) Tensor,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in), optional Header,
    logical, optional blank_at_top,
    logical, optional blank_at_bottom )
```

5.1.1.106 mqc_print_scalar_algebra1()

```
subroutine mqc_algebra::mqc_print_scalar_algebra1 (
    class(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )
```

MQC_Print_Scalar_Algebra1 is a subroutine used to print an **MQC_Scalar**

Purpose:

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar. Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank lines before or after output.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The variable to be printed
in	<i>IOut</i>	IOut is Integer(kind=int64) The Fortran file number to print to
in	<i>Header</i>	Header is Character(Len=*) The title to print along with Scalar
in	<i>Blank_At_Top</i>	Blank_At_Top is Logical,Optional = .True.: print blank line above output = .False.: do not print blank line above output
in	<i>Blank_At_Bottom</i>	Blank_At_Bottom is Logical,Optional = .True.: print blank line below output = .False.: do not print blank line below output

Author

L. M. Thompson

Date

2016

5.1.1.107 mqc_print_vector_algebra1()

```
subroutine mqc_algebra::mqc_print_vector_algebra1 (
    class(mqc_vector), intent(in) Vector,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Verbose,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )
```

5.1.1.108 mqc_r4tensor_at()

```
type(mqc_scalar) function mqc_algebra::mqc_r4tensor_at (
    class(mqc_r4tensor), intent(in) Tensor,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J,
    integer(kind=int64), intent(in) K,
    integer(kind=int64), intent(in) L )
```

5.1.1.109 mqc_r4tensor_havecomplex()

```
logical function mqc_algebra::mqc_r4tensor_havecomplex (
    type(mqc_r4tensor), intent(in) R4Tensor )
```

5.1.1.110 mqc_r4tensor_haveinteger()

```
logical function mqc_algebra::mqc_r4tensor_haveinteger (
    type(mqc_r4tensor), intent(in) R4Tensor )
```

5.1.1.111 mqc_r4tensor_havereal()

```
logical function mqc_algebra::mqc_r4tensor_havereal (
    type(mqc_r4tensor), intent(in) R4Tensor )
```

5.1.1.112 mqc_r4tensor_initialize()

```
subroutine mqc_algebra::mqc_r4tensor_initialize (
    class(mqc_r4tensor), intent(inout) R4Tensor,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J,
    integer(kind=int64), intent(in) K,
    integer(kind=int64), intent(in) L,
    class(*), optional Scalar )
```

5.1.1.113 mqc_r4tensor_put()

```
subroutine mqc_algebra::mqc_r4tensor_put (
    class(mqc_r4tensor), intent(inout) Tensor,
    type(mqc_scalar), intent(in) Element,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in) J,
    integer(kind=int64), intent(in) K,
    integer(kind=int64), intent(in) L )
```

5.1.1.114 mqc_realgtscalar()

```
logical function mqc_algebra::mqc_realgtscalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar

Purpose:

MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real is greater than the real part of the MQC_Scalar and FALSE if the intrinsic real is less than the real part of the MQC_Scalar. If the intrinsic real is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.115 mqc_reallescalar()

```
logical function mqc_algebra::mqc_reallescalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar

Purpose:

MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.116 mqc_realltscalar()

```
logical function mqc_algebra::mqc_realltscalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

Purpose:

MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar. If the intrinsic real is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.117 mqc_realscalaradd()

```
type(mqc_scalar) function mqc_algebra::mqc_realscalaradd (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

5.1.1.118 mqc_realscalardivide()

```
type(mqc_scalar) function mqc_algebra::mqc_realscalardivide (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real variable numerator
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.119 mqc_realscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_realscalarmultiply (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

5.1.1.120 mqc_realscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_realscalarsubtract (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic real

Purpose:

MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic real.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

5.1.1.121 mqc_realvectorproduct()

```
type(mqc_vector) function mqc_algebra::mqc_realvectorproduct (
    real(kind=real64), intent(in) RealIn,
    type(mqc_vector), intent(in) Vector )
```

5.1.1.122 mqc_scalar_acos()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_acos (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar

Purpose:

MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.123 mqc_scalar_asin()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_asin (  
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar

Purpose:

MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.124 mqc_scalar_atan()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_atan (  
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar

Purpose:

MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.125 mqc_scalar_atan2()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_atan2 (  
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram

Purpose:

MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.126 mqc_scalar_cmplx()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_cmplx (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars

Purpose:

MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_Scalar variables.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The real part of MQC_Scalar_Cmplx
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The imaginary part of MQC_Scalar_Cmplx

Author

L. M. Thompson

Date

2019

5.1.1.127 mqc_scalar_complex_conjugate()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_conjugate (
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_Conjugate is a function that returns the complex conjugate of an MQC_Scalar

Purpose:

MQC_Scalar_Complex_Conjugate is a function that returns the complex conjugate of an MQC_Scalar.

Parameters

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable
----	------------------------------	---

Author

L. M. Thompson

Date

2018

5.1.1.128 mqc_scalar_complex_imagpart()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_imagpart (
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_ImagPart is a function that returns the inaginary part of an MQC_Scalar

Purpose:

MQC_Scalar_Complex_RealPart is a function that returns the imaginary part of an MQC_Scalar.

Parameters

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable
----	------------------------------	---

Author

L. M. Thompson

Date

2019

5.1.1.129 mqc_scalar_complex_realpart()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_realpart (
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar

Purpose:

MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar.

Parameters

in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable
----	-----------------	---

Author

L. M. Thompson

Date

2019

5.1.1.130 mqc_scalar_cos()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_cos (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar

Purpose:

MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.131 mqc_scalar_get_abs_value()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_get_abs_value (
    class(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable

Purpose:

MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	--

Author

A. Mahler

Date

2018

5.1.1.132 mqc_scalar_get_intrinsic_complex()

```
complex(kind=real64) function mqc_algebra::mqc_scalar_get_intrinsic_complex (
    class(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Get_Intrinsic_Complex is a function that returns the MQC_scalar value as an intrinsic complex

Purpose:

MQC_Scalar_Get_Intrinsic_Complex is a function that returns the MQC_scalar value as an intrinsic complex.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	--

Author

L. M. Thompson

Date

2017

5.1.1.133 mqc_scalar_get_intrinsic_integer()

```
integer(kind=int64) function mqc_algebra::mqc_scalar_get_intrinsic_integer (
    class(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Get_Intrinsic_Integer is a function that returns the MQC_scalar value as an intrinsic integer

Purpose:

MQC_Scalar_Get_Intrinsic_Integer is a function that returns the MQC_scalar value as an intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	--

Author

L. M. Thompson

Date

2017

5.1.1.134 mqc_scalar_get_intrinsic_real()

```
real(kind=real64) function mqc_algebra::mqc_scalar_get_intrinsic_real (
    class(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Get_Intrinsic_Real is a function that returns the MQC_scalar value as an intrinsic real

Purpose:

MQC_Scalar_Get_Intrinsic_Real is a function that returns the MQC_scalar value as an intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	--

Author

L. M. Thompson

Date

2017

5.1.1.135 mqc_scalar_get_random_value()

```
subroutine mqc_algebra::mqc_scalar_get_random_value (
    class(mqc_scalar) Scalar )
```

MQC_Scalar_Get_Random_Value is a function that returns a random real value from a uniform distribution between zero and one

Purpose:

MQC_Scalar_Get_Random_Value is a function that returns a random real value from a uniform distribution between zero and one.

Parameters

in, out	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be filled
---------	---------------	--

Author

X. Dong

Date

2019

5.1.1.136 mqc_scalar_havecomplex()

```
logical function mqc_algebra::mqc_scalar_havecomplex (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_HaveComplex is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex

Purpose:

MQC_Scalar_HaveComplex is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	---

Author

L. M. Thompson

Date

2017

5.1.1.137 mqc_scalar_haveinteger()

```
logical function mqc_algebra::mqc_scalar_haveinteger (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_HaveInteger is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer

Purpose:

MQC_Scalar_HaveInteger is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	---

Author

L. M. Thompson

Date

2017

5.1.1.138 mqc_scalar_havereal()

```
logical function mqc_algebra::mqc_scalar_havereal (  
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_HaveReal is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real

Purpose:

MQC_Scalar_HaveReal is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested
----	---------------	---

Author

L. M. Thompson

Date

2017

5.1.1.139 `mqc_scalar_isallocated()`

```
logical function mqc_algebra::mqc_scalar_isallocated (
    type(mqc_scalar), intent(inout) Scalar )
```

MQC_Scalar_IsAllocated is used to determine the allocation status of an **MQC_Scalar**

Purpose:

MQC_Scalar_IsAllocated is a subroutine used to determine the allocation status of an MQC_Scalar.

Parameters

in, out	Scalar	Scalar is Type(MQC_Scalar) The name of the MQC_Scalar variable to check allocation status
---------	--------	--

Author

L. M. Thompson

Date

2017

5.1.1.140 `mqc_scalar_sin()`

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_sin (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sin is a function used to return the sine of an **MQC_scalar**

Purpose:

MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.141 mqc_scalar_sqrt()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_sqrt (  
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

Purpose:

MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2016

5.1.1.142 mqc_scalar_tan()

```
type(mqc_scalar) function mqc_algebra::mqc_scalar_tan (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Tan is a function used to return the tangent of an **MQC_scalar**

Purpose:

MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

5.1.1.143 mqc_scalaradd()

```
type(mqc_scalar) function mqc_algebra::mqc_scalaradd (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarAdd is a function that sums two **MQC_Scalar** objects

Purpose:

MQC_ScalarAdd is a function that sums two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be summed
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be summed

Author

L. M. Thompson

Date

2016

5.1.1.144 mqc_scalarcomplexadd()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexadd (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum

Author

L. M. Thompson

Date

2019

5.1.1.145 mqc_scalarcomplexdivide()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexdivide (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex

Purpose:

`MQC_ScalarComplexDivide` is a function that is used to divide an `MQC_Scalar` by an intrinsic complex.

Parameters

in	<i>Scalar</i>	Scalar is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> variable numerator
in	<i>Complex↔ In</i>	<code>ComplexIn</code> is <code>Complex(kind=real64)</code> The intrinsic complex variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.146 mqc_scalarcomplexexponent()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexexponent (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) CompIn )
```

MQC_ScalarComplexExponent is a function that raises an `MQC_Scalar` to the power of an intrinsic complex

Purpose:

`MQC_ScalarComplexExponent` is a function that raises an `MQC_Scalar` to the power of an intrinsic complex.

Parameters

in	<i>Scalar</i>	<code>Scalar1</code> is <code>Type(MQC_Scalar)</code> The base value
in	<i>Comp↔ In</i>	<code>CompIn</code> is <code>Complex(kind=real64)</code> The power value

Author

L. M. Thompson

Date

2019

5.1.1.147 mqc_scalarcomplexmultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexmultiply (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an **MQC_Scalar**

Purpose:

MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply

Author

L. M. Thompson

Date

2019

5.1.1.148 mqc_scalarcomplexsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexsubtract (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an **MQC_Scalar**

Purpose:

`MQC_ScalarComplexSubtract` is a function that is used to subtract an intrinsic complex from an `MQC_Scalar`.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>Complex↔ In</i>	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract

Author

L. M. Thompson

Date

2019

5.1.1.149 mqc_scalardivide()

```
type(mqc_scalar) function mqc_algebra::mqc_scalardivide (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

Purpose:

`MQC_ScalarDivide` is a function that divides `MQC_Scalar` objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The numerator
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The denominator

Author

L. M. Thompson

Date

2016

5.1.1.150 mqc_scalareq()

```
logical function mqc_algebra::mqc_scalareq (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarEQ is a function that returns **TRUE** if two **MQC_Scalar** variables are equal

Purpose:

MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.151 mqc_scalarexponent()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarexponent (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarExponent is a function that raises one **MQC_Scalar** to the power of another **MQC_Scalar**

Purpose:

`MQC_ScalarExponent` is a function that raises one `MQC_Scalar` to the power of another `MQC_Scalar`.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(<code>MQC_Scalar</code>) The base value
in	<i>Scalar2</i>	Scalar2 is Type(<code>MQC_Scalar</code>) The power value

Author

L. M. Thompson

Date

2016

5.1.1.152 mqc_scalarge()

```
logical function mqc_algebra::mqc_scalarge (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGE is a function that returns TRUE if the left `MQC_Scalar` is greater than or equal the right `MQC_Scalar`

Purpose:

`MQC_ScalarGE` is a function that returns TRUE if the left `MQC_Scalar` is greater than or equal to the right `MQC_Scalar`.

When dealing with complex numbers, the function returns TRUE if the left real part is is greater than or equal to the right real part and FALSE if the left real part is less than the right real part.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(<code>MQC_Scalar</code>) The first <code>MQC_Scalar</code> that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(<code>MQC_Scalar</code>) The second <code>MQC_Scalar</code> that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.153 mqc_scalargt()

```
logical function mqc_algebra::mqc_scalargt (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGT is a function that returns **TRUE** if the left **MQC_Scalar** is greater than the right **MQC_Scalar**

Purpose:

MQC_ScalarGT is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is greater than the right real part and FALSE if the left real part is less than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is greater than the right imaginary part and FALSE otherwise.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.154 `mqc_scalargtinteger()`

```
logical function mqc_algebra::mqc_scalargtinteger (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer

Purpose:

MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is greater than the intrinsic integer and FALSE if the real part of the MQC_Scalar is less than the intrinsic integer. If the real part of the MQC_Scalar is equal to the intrinsic integer, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.155 `mqc_scalargtreall()`

```
logical function mqc_algebra::mqc_scalargtreall (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real

Purpose:

`MQC_ScalarGTReal` is a function that returns `TRUE` if a `MQC_Scalar` is greater than an intrinsic real.

When dealing with complex numbers, the function returns `TRUE` if the real part of the `MQC_Scalar` is greater than the intrinsic real and `FALSE` if the real part of the `MQC_Scalar` is less than the intrinsic real. If the real part of the `MQC_Scalar` is equal to the intrinsic real, the function returns `TRUE` if the imaginary part of `MQC_Scalar` is greater than zero and `FALSE` otherwise.

Parameters

in	<i>Scalar</i>	Scalar is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> that will be tested.
in	<i>IntIn</i>	<code>RealIn</code> is <code>Real(kind=int64)</code> The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.156 mqc_scalarintegeradd()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarintegeradd (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an `MQC_Scalar`

Purpose:

`MQC_ScalarIntegerSum` is a function that is used to sum an intrinsic integer by an `MQC_Scalar`.

Parameters

in	<i>Integer↔ In</i>	<code>IntegerIn</code> is <code>Integer(kind=int64)</code> The intrinsic integer variable to sum
in	<i>Scalar</i>	
Generated by Doxygen		<code>Scalar</code> is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> varibale to sum

Author

L. M. Thompson

Date

2019

5.1.1.157 mqc_scalarintegerdivide()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarintegerdivide (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerDivide is a function that is used to divide an **MQC_Scalar** by an intrinsic integer

Purpose:

MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator
in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.158 mqc_scalarintegerexponent()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarintegerexponent (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarIntegerExponent is a function that raises an **MQC_Scalar** to the power of an intrinsic integer

Purpose:

`MQC_ScalarIntegerExponent` is a function that raises an `MQC_Scalar` to the power of an intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The power value

Author

L. M. Thompson

Date

2019

5.1.1.159 mqc_scalarintegermultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarintegermultiply (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an `MQC_Scalar`

Purpose:

`MQC_ScalarIntegerMultiply` is a function that is used to multiply an intrinsic integer by an `MQC_Scalar`.

Parameters

in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

5.1.1.160 mqc_scalarintegersubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarintegersubtract (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar

Purpose:

MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract

Author

L. M. Thompson

Date

2019

5.1.1.161 mqc_scalarle()

```
logical function mqc_algebra::mqc_scalarle (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar

Purpose:

MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal to the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is less than or equal to the right real part and FALSE if the left real part is greater than the right real part.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.162 mqc_scalarleinteger()

```
logical function mqc_algebra::mqc_scalarleinteger (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer

Purpose:

MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than or equal to the intrinsic integer and FALSE if the real part of the MQC_Scalar is greater than the intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.163 mqc_scalarlereal()

```
logical function mqc_algebra::mqc_scalarlereal (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLereal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real

Purpose:

MQC_ScalarLereal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than or equal to the intrinsic real and FALSE if the real part of the MQC_Scalar is greater than the intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	RealIn is Real(kind=int64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.164 mqc_scalarlt()

```
logical function mqc_algebra::mqc_scalarlt (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLT is a function that returns **TRUE** if the left **MQC_Scalar** is less than the right **MQC_Scalar**

Purpose:

MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is less than the right real part and FALSE if the left real part is greater than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is less than the right imaginary part and FALSE otherwise.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.165 mqc_scalarltreal()

```
logical function mqc_algebra::mqc_scalarltreal (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real

Purpose:

MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than the intrinsic real and FALSE if the real part of the MQC_Scalar is greater than the intrinsic real. If the real part of the MQC_Scalar is equal to the intrinsic real, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

5.1.1.166 mqc_scalarmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::mqc_scalarmatrixproduct (
    type(mqc_scalar), intent(in) Scalar,
    type(mqc_matrix), intent(in) Matrix )
```

5.1.1.167 mqc_scalarmultiply()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarmultiply (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

Purpose:

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be multiplied
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be multiplied

Author

L. M. Thompson

Date

2016

5.1.1.168 mqc_scalarne()

```
logical function mqc_algebra::mqc_scalarne (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal

Purpose:

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.1.169 mqc_scalarrealadd()

```

type(mqc_scalar) function mqc_algebra::mqc_scalarrealadd (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )

```

MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

MQC_ScalarRealSum is a function that is used to sum an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

5.1.1.170 mqc_scalarrealdive()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarrealdive (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealDive is a function that is used to divide an MQC_Scalar by an intrinsic real

Purpose:

MQC_ScalarRealDive is a function that is used to divide an MQC_Scalar by an intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator
in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real variable denominator

Author

L. M. Thompson

Date

2019

5.1.1.171 mqc_scalarrealexponent()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarrealexponent (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real

Purpose:

MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>RealIn</i>	RealIn is Real(kind=real64) The power value

Author

L. M. Thompson

Date

2019

5.1.1.172 mqc_scalarrealmultiply()

```

type(mqc_scalar) function mqc_algebra::mqc_scalarrealmultiply (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )

```

MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

5.1.1.173 mqc_scalarrealsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarrealsubtract (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar

Purpose:

MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real to subtract

Author

L. M. Thompson

Date

2019

5.1.1.174 mqc_scalarsubtract()

```
type(mqc_scalar) function mqc_algebra::mqc_scalarsubtract (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

Purpose:

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar from which Scalar2 will be subtracted
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar which will be subtracted from Scalar1

Author

L. M. Thompson

Date

2016

5.1.1.175 mqc_scalarvectordifference()

```
type(mqc_vector) function mqc_algebra::mqc_scalarvectordifference (
    type(mqc_scalar), intent(in) ScalarIn,
    type(mqc_vector), intent(in) VectorIn )
```

5.1.1.176 mqc_scalarvectorproduct()

```
type(mqc_vector) function mqc_algebra::mqc_scalarvectorproduct (
    type(mqc_scalar), intent(in) Scalar,
    type(mqc_vector), intent(in) Vector )
```

5.1.1.177 mqc_scalarvectorsum()

```
type(mqc_vector) function mqc_algebra::mqc_scalarvectorsum (
    type(mqc_scalar), intent(in) ScalarIn,
    type(mqc_vector), intent(in) VectorIn )
```


5.1.1.178 mqc_set_array2tensor()

```
subroutine mqc_algebra::mqc_set_array2tensor (
    type(mqc_r4tensor), intent(inout) TensorOut,
    class(*), dimension(:,:,:), intent(in) ArrayIn )
```

5.1.1.179 mqc_set_array2vector_complex()

```
subroutine mqc_algebra::mqc_set_array2vector_complex (
    type(mqc_vector), intent(inout) VectorOut,
    complex(kind=real64), dimension(:), intent(in) ArrayIn )
```

5.1.1.180 mqc_set_array2vector_integer()

```
subroutine mqc_algebra::mqc_set_array2vector_integer (
    type(mqc_vector), intent(inout) VectorOut,
    integer(kind=int64), dimension(:), intent(in) ArrayIn )
```

5.1.1.181 mqc_set_array2vector_real()

```
subroutine mqc_algebra::mqc_set_array2vector_real (
    type(mqc_vector), intent(inout) VectorOut,
    real(kind=real64), dimension(:), intent(in) ArrayIn )
```

5.1.1.182 mqc_set_complexarray2matrix()

```
subroutine mqc_algebra::mqc_set_complexarray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    complex(kind=real64), dimension(:,:), intent(in) ArrayIn )
```

5.1.1.183 mqc_set_integerarray2matrix()

```
subroutine mqc_algebra::mqc_set_integerarray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    integer(kind=int64), dimension(:,:), intent(in) ArrayIn )
```

5.1.1.184 mqc_set_matrix2complexarray()

```
subroutine mqc_algebra::mqc_set_matrix2complexarray (
    complex(kind=real64), dimension(:,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

5.1.1.185 mqc_set_matrix2integerarray()

```
subroutine mqc_algebra::mqc_set_matrix2integerarray (
    integer(kind=int64), dimension(:,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

5.1.1.186 mqc_set_matrix2matrix()

```
subroutine mqc_algebra::mqc_set_matrix2matrix (
    class(mqc_matrix), intent(inout) MatrixOut,
    class(mqc_matrix), intent(in) MatrixIn )
```

5.1.1.187 mqc_set_matrix2realarray()

```
subroutine mqc_algebra::mqc_set_matrix2realarray (
    real(kind=real64), dimension(:,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

5.1.1.188 mqc_set_rearray2matrix()

```
subroutine mqc_algebra::mqc_set_rearray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    real(kind=real64), dimension(:,:), intent(in) ArrayIn )
```

5.1.1.189 mqc_set_vector2complexarray()

```
subroutine mqc_algebra::mqc_set_vector2complexarray (
    complex(kind=real64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```

5.1.1.190 mqc_set_vector2integerarray()

```
subroutine mqc_algebra::mqc_set_vector2integerarray (
    integer(kind=int64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```

5.1.1.191 mqc_set_vector2realarray()

```
subroutine mqc_algebra::mqc_set_vector2realarray (
    real(kind=real64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```

5.1.1.192 mqc_set_vector2vector()

```
subroutine mqc_algebra::mqc_set_vector2vector (
    class(mqc_vector), intent(inout) VectorOut,
    class(mqc_vector), intent(in) VectorIn )
```

5.1.1.193 mqc_vector2diagmatrix()

```
type(mqc_matrix) function mqc_algebra::mqc_vector2diagmatrix (
    type(mqc_vector), intent(in) vector )
```

5.1.1.194 mqc_vector_abs()

```
type(mqc_vector) function mqc_algebra::mqc_vector_abs (
    class(mqc_vector), intent(in) A )
```

5.1.1.195 mqc_vector_argsort()

```
type(mqc_vector) function mqc_algebra::mqc_vector_argsort (
    class(mqc_vector), intent(in) Vector )
```

5.1.1.196 mqc_vector_cast_complex()

```
type(mqc_vector) function mqc_algebra::mqc_vector_cast_complex (
    type(mqc_vector), intent(in) VA )
```

5.1.1.197 mqc_vector_cast_real()

```
type(mqc_vector) function mqc_algebra::mqc_vector_cast_real (
    type(mqc_vector), intent(in) VA )
```

5.1.1.198 mqc_vector_cmplx()

```
type(mqc_vector) function mqc_algebra::mqc_vector_cmplx (
    type(mqc_vector), intent(in) Vector1,
    type(mqc_vector), intent(in) Vector2 )
```

5.1.1.199 mqc_vector_complex_imagpart()

```
type(mqc_vector) function mqc_algebra::mqc_vector_complex_imagpart (
    class(mqc_vector), intent(in) A )
```

5.1.1.200 mqc_vector_complex_realpart()

```
type(mqc_vector) function mqc_algebra::mqc_vector_complex_realpart (
    class(mqc_vector), intent(in) A )
```

5.1.1.201 mqc_vector_conjugate_transpose()

```
type(mqc_vector) function mqc_algebra::mqc_vector_conjugate_transpose (
    class(mqc_vector), intent(in) Vector )
```

5.1.1.202 mqc_vector_copy_complex2int()

```
subroutine mqc_algebra::mqc_vector_copy_complex2int (  
    type(mqc_vector) Vector )
```

5.1.1.203 mqc_vector_copy_complex2real()

```
subroutine mqc_algebra::mqc_vector_copy_complex2real (  
    type(mqc_vector) Vector )
```

5.1.1.204 mqc_vector_copy_int2complex()

```
subroutine mqc_algebra::mqc_vector_copy_int2complex (  
    type(mqc_vector) Vector )
```

5.1.1.205 mqc_vector_copy_int2real()

```
subroutine mqc_algebra::mqc_vector_copy_int2real (  
    type(mqc_vector) Vector )
```

5.1.1.206 mqc_vector_copy_real2complex()

```
subroutine mqc_algebra::mqc_vector_copy_real2complex (  
    type(mqc_vector) Vector )
```

5.1.1.207 mqc_vector_copy_real2int()

```
subroutine mqc_algebra::mqc_vector_copy_real2int (  
    type(mqc_vector) Vector )
```

5.1.1.208 mqc_vector_havecomplex()

```
logical function mqc_algebra::mqc_vector_havecomplex (
    type(mqc_vector), intent(in) Vector )
```

5.1.1.209 mqc_vector_haveinteger()

```
logical function mqc_algebra::mqc_vector_haveinteger (
    type(mqc_vector), intent(in) Vector )
```

5.1.1.210 mqc_vector_havereal()

```
logical function mqc_algebra::mqc_vector_havereal (
    type(mqc_vector), intent(in) Vector )
```

5.1.1.211 mqc_vector_initialize()

```
subroutine mqc_algebra::mqc_vector_initialize (
    class(mqc_vector), intent(inout) Vector,
    integer(kind=int64), intent(in) Length,
    class(*), optional Scalar )
```

5.1.1.212 mqc_vector_isallocated()

```
logical function mqc_algebra::mqc_vector_isallocated (
    class(mqc_vector), intent(inout) Vector )
```

5.1.1.213 mqc_vector_iscolumn()

```
logical function mqc_algebra::mqc_vector_iscolumn (
    type(mqc_vector), intent(in) Vector )
```

5.1.1.214 mqc_vector_maxloc()

```
integer function mqc_algebra::mqc_vector_maxloc (  
    class(mqc_vector), intent(in) Vector )
```

5.1.1.215 mqc_vector_maxval()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_maxval (  
    class(mqc_vector), intent(in) Vector )
```

5.1.1.216 mqc_vector_minloc()

```
integer function mqc_algebra::mqc_vector_minloc (  
    class(mqc_vector), intent(in) Vector )
```

5.1.1.217 mqc_vector_minval()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_minval (  
    class(mqc_vector), intent(in) Vector )
```

5.1.1.218 mqc_vector_norm()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_norm (  
    class(mqc_vector), intent(inout) vector,  
    character(len=1), intent(in), optional methodIn )
```

5.1.1.219 mqc_vector_pop()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_pop (  
    class(mqc_vector), intent(inout) Vector )
```

5.1.1.220 mqc_vector_power()

```
subroutine mqc_algebra::mqc_vector_power (
    class(mqc_vector), intent(inout) A,
    class(*) P )
```

5.1.1.221 mqc_vector_push()

```
subroutine mqc_algebra::mqc_vector_push (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_scalar), intent(in) Scalar )
```

5.1.1.222 mqc_vector_scalar_at()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_scalar_at (
    class(mqc_vector) Vec,
    integer(kind=int64), intent(in) I )
```

5.1.1.223 mqc_vector_scalar_increment()

```
subroutine mqc_algebra::mqc_vector_scalar_increment (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) I )
```

5.1.1.224 mqc_vector_scalar_put()

```
subroutine mqc_algebra::mqc_vector_scalar_put (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) I )
```

5.1.1.225 mqc_vector_shift()

```
type(mqc_scalar) function mqc_algebra::mqc_vector_shift (
    class(mqc_vector), intent(inout) Vector )
```


5.1.1.226 mqc_vector_sort()

```
subroutine mqc_algebra::mqc_vector_sort (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_vector), intent(in), optional idx )
```

5.1.1.227 mqc_vector_sqrt()

```
subroutine mqc_algebra::mqc_vector_sqrt (
    class(mqc_vector), intent(inout) A )
```

5.1.1.228 mqc_vector_transpose()

```
type(mqc_vector) function mqc_algebra::mqc_vector_transpose (
    class(mqc_vector), intent(in) Vector )
```

5.1.1.229 mqc_vector_unshift()

```
subroutine mqc_algebra::mqc_vector_unshift (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_scalar), intent(in) Scalar )
```

5.1.1.230 mqc_vector_vector_at()

```
type(mqc_vector) function mqc_algebra::mqc_vector_vector_at (
    class(mqc_vector) Vec,
    integer(kind=int64), intent(in) I,
    integer(kind=int64), intent(in), optional J )
```

5.1.1.231 mqc_vector_vector_put()

```
subroutine mqc_algebra::mqc_vector_vector_put (
    class(mqc_vector), intent(inout) Vector,
    type(mqc_vector), intent(in) VectorIn,
    integer(kind=int64), intent(in), optional I )
```

5.1.1.232 mqc_vectorcomplexdivide()

```
type(mqc_vector) function mqc_algebra::mqc_vectorcomplexdivide (
    type(mqc_vector), intent(in) vector,
    complex(kind=real64), intent(in) compIn )
```

5.1.1.233 mqc_vectorcomplexproduct()

```
type(mqc_vector) function mqc_algebra::mqc_vectorcomplexproduct (
    type(mqc_vector), intent(in) vector,
    complex(kind=real64), intent(in) compIn )
```

5.1.1.234 mqc_vectorintegerdivide()

```
type(mqc_vector) function mqc_algebra::mqc_vectorintegerdivide (
    type(mqc_vector), intent(in) vector,
    integer(kind=int64), intent(in) intIn )
```

5.1.1.235 mqc_vectorintegerproduct()

```
type(mqc_vector) function mqc_algebra::mqc_vectorintegerproduct (
    type(mqc_vector), intent(in) vector,
    integer(kind=int64), intent(in) intIn )
```

5.1.1.236 mqc_vectormatrixdotproduct()

```
type(mqc_vector) function mqc_algebra::mqc_vectormatrixdotproduct (
    type(mqc_vector), intent(in) VA,
    type(mqc_matrix), intent(in) MB )
```

5.1.1.237 mqc_vectorrealddivide()

```
type(mqc_vector) function mqc_algebra::mqc_vectorrealddivide (
    type(mqc_vector), intent(in) vector,
    real(kind=real64), intent(in) realIn )
```

5.1.1.238 mqc_vectorrealproduct()

```
type(mqc_vector) function mqc_algebra::mqc_vectorrealproduct (
    type(mqc_vector), intent(in) vector,
    real(kind=real64), intent(in) realIn )
```

5.1.1.239 mqc_vectorscalardivide()

```
type(mqc_vector) function mqc_algebra::mqc_vectorscalardivide (
    type(mqc_vector), intent(in) vector,
    type(mqc_scalar), intent(in) scalar )
```

5.1.1.240 mqc_vectorscalarproduct()

```
type(mqc_vector) function mqc_algebra::mqc_vectorscalarproduct (
    type(mqc_vector), intent(in) vector,
    type(mqc_scalar), intent(in) scalar )
```

5.1.1.241 mqc_vectorvectordifference()

```
type(mqc_vector) function mqc_algebra::mqc_vectorvectordifference (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

5.1.1.242 mqc_vectorvectordotproduct()

```
type(mqc_scalar) function mqc_algebra::mqc_vectorvectordotproduct (
    type(mqc_vector), intent(in) Vector1,
    type(mqc_vector), intent(in) Vector2 )
```

5.1.1.243 mqc_vectorvectorsum()

```
type(mqc_vector) function mqc_algebra::mqc_vectorvectorsum (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

5.1.1.244 symindexhash()

```
integer(kind=int64) function mqc_algebra::symindexhash (
    integer(kind=int64), intent(in) i,
    integer(kind=int64), intent(in) j,
    integer(kind=int64), intent(in), optional k,
    integer(kind=int64), intent(in), optional l )
```

5.2 mqc_est Module Reference

Data Types

- interface [assignment\(=\)](#)
- interface [contraction](#)
- interface [dagger](#)
- interface [dot_product](#)
- interface [matmul](#)
- type [mqc_determinant](#)
- type [mqc_determinant_string](#)
- interface [mqc_matrix_undospinblockghf](#)
- interface [mqc_print](#)
- type [mqc_pscf_wavefunction](#)
- type [mqc_scf_eigenvalues](#)
- type [mqc_scf_integral](#)
- type [mqc_twoeris](#)
- type [mqc_wavefunction](#)
- interface [operator\(*\)](#)
- interface [operator\(+\)](#)
- interface [operator\(-\)](#)
- interface [transpose](#)

Functions/Subroutines

- subroutine [mqc_print_wavefunction](#) (wavefunction, iOut, label)
- subroutine [mqc_print_integral](#) (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_eigenvalues](#) (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_twoeris](#) (twoERIs, iOut, header, blank_at_top, blank_at_bottom)
- logical function [mqc_integral_isallocated](#) (Integral)
- logical function [mqc_eigenvalues_isallocated](#) (Eigenvalues)
- logical function [mqc_integral_has_alpha](#) (integral)
- logical function [mqc_integral_has_beta](#) (integral)
- logical function [mqc_integral_has_alphabeta](#) (integral)
- logical function [mqc_integral_has_betaalpha](#) (integral)
- logical function [mqc_eigenvalues_has_alpha](#) (eigenvalues)
- logical function [mqc_eigenvalues_has_beta](#) (eigenvalues)
- character(len=64) function [mqc_integral_array_type](#) (integral)
- character(len=64) function [mqc_eigenvalues_array_type](#) (eigenvalues)

- character(len=64) function [mqc_integral_array_name](#) (integral)
- character(len=64) function [mqc_eigenvalues_array_name](#) (eigenvalues)
- subroutine [mqc_integral_add_name](#) (integral, arrayName)
- subroutine [mqc_eigenvalues_add_name](#) (eigenvalues, arrayName)
- integer(kind=int64) function [mqc_integral_dimension](#) (integral, label, axis)
- integer(kind=int64) function [mqc_eigenvalues_dimension](#) (eigenvalues, label)
- subroutine [mqc_twoeris_allocate](#) (twoERIs, storageType, integralType, alpha, beta, alphaBeta, betaAlpha)
- subroutine [mqc_integral_allocate](#) (integral, arrayName, arrayType, alpha, beta, alphaBeta, betaAlpha)
- subroutine [mqc_eigenvalues_allocate](#) (eigenvalues, arrayName, arrayType, alpha, beta)
- subroutine [mqc_integral_identity](#) (integral, nAlpha, nBeta, label, nAlpha2, nBeta2)
- subroutine [mqc_integral_initialize](#) (integral, nAlpha, nBeta, scalar, label, nAlpha2, nBeta2)
- type(mqc_matrix) function [mqc_integral_output_block](#) (integral, blockName)
- type(mqc_scf_integral) function [mqc_integral_output_orbitals](#) (integral, orbString, alphaOrbsIn, betaOrbsIn, axis)
- type(mqc_scf_integral) function [mqc_integral_swap_orbitals](#) (integral, alphaOrbsIn, betaOrbsIn, axis)
- type(mqc_vector) function [mqc_eigenvalues_output_block](#) (eigenvalues, blockName)
- subroutine [mqc_integral_output_array](#) (matrixOut, integralln)
- subroutine [mqc_eigenvalues_output_array](#) (vectorOut, eigenvaluesIn)
- type(mqc_scf_integral) function [mqc_integral_matrix_multiply](#) (integralA, matrixB, label)
- type(mqc_scf_integral) function [mqc_matrix_integral_multiply](#) (matrixA, integralB, label)
- type(mqc_scf_integral) function [mqc_integral_sum](#) (integralA, integralB)
- type(mqc_scf_integral) function [mqc_integral_difference](#) (integralA, integralB)
- type(mqc_scf_integral) function [mqc_integral_integral_multiply](#) (integralA, integralB, label)
- type(mqc_scf_integral) function [mqc_scalar_integral_multiply](#) (scalar, integral)
- type(mqc_scf_integral) function [mqc_integral_scalar_multiply](#) (integral, scalar)
- type(mqc_scf_integral) function [mqc_integral_eigenvalues_multiply](#) (integralA, eigenvaluesB, label)
- type(mqc_scf_integral) function [mqc_eigenvalues_integral_multiply](#) (eigenvaluesA, integralB, label)
- type(mqc_scf_eigenvalues) function [mqc_eigenvalues_eigenvalues_multiply](#) (eigenvaluesA, eigenvaluesB, label)
- type(mqc_scalar) function [mqc_eigenvalue_eigenvalue_dotproduct](#) (eigenvalueA, eigenvalueB)
- type(mqc_scf_integral) function [mqc_integral_transpose](#) (integral, label)
- type(mqc_scf_integral) function [mqc_integral_conjugate_transpose](#) (integral, label)
- type(mqc_scalar) function [mqc_integral_norm](#) (integral, methodIn)
- subroutine [mqc_matrix_spinblockghf](#) (array, nelec, multi, elist)
- subroutine [mqc_matrix_undospinblockghf_eigenvalues](#) (eigenvaluesIn, vectorOut)
- subroutine [mqc_matrix_undospinblockghf_integral](#) (integralln, matrixOut)
- type(mqc_scalar) function [mqc_scf_integral_contraction](#) (integral1, integral2)
- type(mqc_scf_integral) function [mqc_eri_integral_contraction](#) (eris, integral, label)
- subroutine [mqc_scf_integral_generalized_eigensystem](#) (integralA, integralB, eVals, rEVecs, IEVecs)
- subroutine [mqc_scf_integral_diagonalize](#) (integral, eVals, eVecs)
- type(mqc_scf_integral) function [mqc_scf_integral_inverse](#) (integral)
- type(mqc_scalar) function [mqc_scf_integral_trace](#) (integral)
- type(mqc_scalar) function [mqc_scf_integral_determinant](#) (integral)
- subroutine [mqc_integral_set_energy_list](#) (integral, elist)
- integer(kind=int64) function, dimension(:), allocatable [mqc_integral_get_energy_list](#) (integral)
- subroutine [mqc_integral_delete_energy_list](#) (integral)
- subroutine [mqc_scf_eigenvalues_power](#) (eigenvalues, power)
- type(mqc_scalar) function [mqc_twoeris_at](#) (twoERIs, i, j, k, l, spinBlock)
- type(mqc_scalar) function [mqc_integral_at](#) (integral, i, j, spinBlock)
- type(mqc_scalar) function [mqc_eigenvalues_at](#) (eigenvalues, i, spinBlock)
- subroutine [mqc_scf_transformation_matrix](#) (overlap, transform_matrix, nBasUse)
- subroutine [gen_det_str](#) (IOut, IPrint, NBasisIn, NAlphaIn, NBetaIn, Determinants, NCoreIn)

- type(mqc_scalar) function [slater_condon](#) (IOut, IPrint, NBasisIn, Determinants, L_A_String, L_B_String, R_A_String, R_B_String, Core_Hamiltonian, ERIs, UHF)
- subroutine [twoeri_trans](#) (IOut, IPrint, MO_Coeff, ERIs, MO_ERIs, UHF)
- subroutine [mqc_build_ci_hamiltonian](#) (IOut, IPrint, NBasis, Determinants, MO_Core_Ham, MO_ERIs, UHF, CI_Hamiltonian)
- type(mqc_matrix) function [get_one_gamma_matrix](#) (iOut, iPrint, nBasisIn, nState, determinants, ci_amplitudes, nCoreIn, nOrbsIn)

5.2.1 Function/Subroutine Documentation

5.2.1.1 gen_det_str()

```
subroutine mqc_est::gen_det_str (
    integer(kind=int64) IOut,
    integer(kind=int64) IPrint,
    type(mqc_scalar) NBasisIn,
    type(mqc_scalar) NAlphaIn,
    type(mqc_scalar) NBetaIn,
    type(mqc_determinant) Determinants,
    type(mqc_scalar), optional NCoreIn )
```

5.2.1.2 get_one_gamma_matrix()

```
type(mqc_matrix) function mqc_est::get_one_gamma_matrix (
    integer(kind=int64), intent(in) iOut,
    integer(kind=int64), intent(in) iPrint,
    type(mqc_scalar), intent(in) nBasisIn,
    integer(kind=int64), intent(in) nState,
    type(mqc_determinant), intent(in) determinants,
    type(mqc_matrix), intent(in) ci_amplitudes,
    integer(kind=int64), intent(in), optional nCoreIn,
    integer(kind=int64), intent(in), optional nOrbsIn )
```

5.2.1.3 mqc_build_ci_hamiltonian()

```
subroutine mqc_est::mqc_build_ci_hamiltonian (
    integer(kind=int64), intent(in) IOut,
    integer(kind=int64), intent(in) IPrint,
    type(mqc_scalar), intent(in) NBasis,
    type(mqc_determinant), intent(in) Determinants,
    type(mqc_scf_integral), intent(in) MO_Core_Ham,
    type(mqc_twoeris), intent(in) MO_ERIs,
    logical, intent(in) UHF,
    type(mqc_matrix), intent(out) CI_Hamiltonian )
```

5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()

```
type(mqc_scalar) function mqc_est::mqc_eigenvalue_eigenvalue_dotproduct (
    type(mqc_scf_eigenvalues), intent(in) eigenvalueA,
    type(mqc_scf_eigenvalues), intent(in) eigenvalueB )
```

5.2.1.5 mqc_eigenvalues_add_name()

```
subroutine mqc_est::mqc_eigenvalues_add_name (
    class(mqc_scf_eigenvalues) eigenvalues,
    character(len=*) arrayName )
```

5.2.1.6 mqc_eigenvalues_allocate()

```
subroutine mqc_est::mqc_eigenvalues_allocate (
    class(mqc_scf_eigenvalues) eigenvalues,
    character(len=*) arrayName,
    character(len=*) arrayType,
    type(mqc_vector), optional alpha,
    type(mqc_vector), optional beta )
```

5.2.1.7 mqc_eigenvalues_array_name()

```
character(len=64) function mqc_est::mqc_eigenvalues_array_name (
    class(mqc_scf_eigenvalues) eigenvalues )
```

5.2.1.8 mqc_eigenvalues_array_type()

```
character(len=64) function mqc_est::mqc_eigenvalues_array_type (
    class(mqc_scf_eigenvalues) eigenvalues )
```

5.2.1.9 mqc_eigenvalues_at()

```
type(mqc_scalar) function mqc_est::mqc_eigenvalues_at (
    class(mqc_scf_eigenvalues) eigenvalues,
    integer(kind=int64) i,
    character(len=64), optional spinBlock )
```

5.2.1.10 mqc_eigenvalues_dimension()

```
integer(kind=int64) function mqc_est::mqc_eigenvalues_dimension (  
    class(mqc_scf_eigenvalues), intent(in) eigenvalues,  
    character(len=*), intent(in) label )
```

5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()

```
type(mqc_scf_eigenvalues) function mqc_est::mqc_eigenvalues_eigenvalues_multiply (  
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesA,  
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesB,  
    character(len=*), intent(in), optional label )
```

5.2.1.12 mqc_eigenvalues_has_alpha()

```
logical function mqc_est::mqc_eigenvalues_has_alpha (  
    class(mqc_scf_eigenvalues) eigenvalues )
```

5.2.1.13 mqc_eigenvalues_has_beta()

```
logical function mqc_est::mqc_eigenvalues_has_beta (  
    class(mqc_scf_eigenvalues) eigenvalues )
```

5.2.1.14 mqc_eigenvalues_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_eigenvalues_integral_multiply (  
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesA,  
    type(mqc_scf_integral), intent(in) integralB,  
    character(len=*), intent(in), optional label )
```

5.2.1.15 mqc_eigenvalues_isallocated()

```
logical function mqc_est::mqc_eigenvalues_isallocated (  
    class(mqc_scf_eigenvalues), intent(inout) Eigenvalues )
```


5.2.1.16 mqc_eigenvalues_output_array()

```
subroutine mqc_est::mqc_eigenvalues_output_array (
    type(mqc_vector), intent(inout) vectorOut,
    class(mqc_scf_eigenvalues), intent(in) eigenvaluesIn )
```

5.2.1.17 mqc_eigenvalues_output_block()

```
type(mqc_vector) function mqc_est::mqc_eigenvalues_output_block (
    class(mqc_scf_eigenvalues) eigenvalues,
    character(len=*), optional blockName )
```

5.2.1.18 mqc_eri_integral_contraction()

```
type(mqc_scf_integral) function mqc_est::mqc_eri_integral_contraction (
    type(mqc_twoeris), intent(in) eris,
    type(mqc_scf_integral), intent(in) integral,
    character(len=*), optional label )
```

5.2.1.19 mqc_integral_add_name()

```
subroutine mqc_est::mqc_integral_add_name (
    class(mqc_scf_integral) integral,
    character(len=*) arrayName )
```

5.2.1.20 mqc_integral_allocate()

```
subroutine mqc_est::mqc_integral_allocate (
    class(mqc_scf_integral) integral,
    character(len=*) arrayName,
    character(len=*) arrayType,
    type(mqc_matrix), optional alpha,
    type(mqc_matrix), optional beta,
    type(mqc_matrix), optional alphaBeta,
    type(mqc_matrix), optional betaAlpha )
```

5.2.1.21 mqc_integral_array_name()

```
character(len=64) function mqc_est::mqc_integral_array_name (
    class(mqc_scf_integral) integral )
```

5.2.1.22 mqc_integral_array_type()

```
character(len=64) function mqc_est::mqc_integral_array_type (
    class(mqc_scf_integral) integral )
```

5.2.1.23 mqc_integral_at()

```
type(mqc_scalar) function mqc_est::mqc_integral_at (
    class(mqc_scf_integral) integral,
    integer(kind=int64) i,
    integer(kind=int64) j,
    character(len=64), optional spinBlock )
```

5.2.1.24 mqc_integral_conjugate_transpose()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_conjugate_transpose (
    type(mqc_scf_integral), intent(in) integral,
    character(len=*), intent(in), optional label )
```

5.2.1.25 mqc_integral_delete_energy_list()

```
subroutine mqc_est::mqc_integral_delete_energy_list (
    class(mqc_scf_integral) integral )
```

5.2.1.26 mqc_integral_difference()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_difference (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), intent(in) integralB )
```

5.2.1.27 mqc_integral_dimension()

```
integer(kind=int64) function mqc_est::mqc_integral_dimension (  
    class(mqc_scf_integral), intent(in) integral,  
    character(len=*), intent(in) label,  
    integer(kind=int64), intent(in), optional axis )
```

5.2.1.28 mqc_integral_eigenvalues_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_eigenvalues_multiply (  
    type(mqc_scf_integral), intent(in) integralA,  
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesB,  
    character(len=*), intent(in), optional label )
```

5.2.1.29 mqc_integral_get_energy_list()

```
integer(kind=int64) function, dimension(:), allocatable mqc_est::mqc_integral_get_energy_list (  
    class(mqc_scf_integral) integral )
```

5.2.1.30 mqc_integral_has_alpha()

```
logical function mqc_est::mqc_integral_has_alpha (  
    class(mqc_scf_integral) integral )
```

5.2.1.31 mqc_integral_has_alphabeta()

```
logical function mqc_est::mqc_integral_has_alphabeta (  
    class(mqc_scf_integral) integral )
```

5.2.1.32 mqc_integral_has_beta()

```
logical function mqc_est::mqc_integral_has_beta (  
    class(mqc_scf_integral) integral )
```

5.2.1.33 mqc_integral_has_betaalpha()

```
logical function mqc_est::mqc_integral_has_betaalpha (
    class(mqc_scf_integral) integral )
```

5.2.1.34 mqc_integral_identity()

```
subroutine mqc_est::mqc_integral_identity (
    class(mqc_scf_integral), intent(inout) integral,
    integer, intent(in) nAlpha,
    integer, intent(in) nBeta,
    character(len=*), intent(in), optional label,
    integer, intent(in), optional nAlpha2,
    integer, intent(in), optional nBeta2 )
```

5.2.1.35 mqc_integral_initialize()

```
subroutine mqc_est::mqc_integral_initialize (
    class(mqc_scf_integral), intent(inout) integral,
    integer, intent(in) nAlpha,
    integer, intent(in) nBeta,
    class(*), intent(in), optional scalar,
    character(len=*), intent(in), optional label,
    integer, intent(in), optional nAlpha2,
    integer, intent(in), optional nBeta2 )
```

5.2.1.36 mqc_integral_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_integral_multiply (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), intent(in) integralB,
    character(len=*), intent(in), optional label )
```

5.2.1.37 mqc_integral_isallocated()

```
logical function mqc_est::mqc_integral_isallocated (
    class(mqc_scf_integral), intent(inout) Integral )
```

5.2.1.38 mqc_integral_matrix_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_matrix_multiply (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_matrix), intent(in) matrixB,
    character(len=*), intent(in), optional label )
```

5.2.1.39 mqc_integral_norm()

```
type(mqc_scalar) function mqc_est::mqc_integral_norm (
    class(mqc_scf_integral), intent(in) integral,
    character(len=1), intent(in), optional methodIn )
```

5.2.1.40 mqc_integral_output_array()

```
subroutine mqc_est::mqc_integral_output_array (
    type(mqc_matrix), intent(inout) matrixOut,
    class(mqc_scf_integral), intent(in) integralIn )
```

5.2.1.41 mqc_integral_output_block()

```
type(mqc_matrix) function mqc_est::mqc_integral_output_block (
    class(mqc_scf_integral) integral,
    character(len=*), optional blockName )
```

5.2.1.42 mqc_integral_output_orbitals()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_output_orbitals (
    class(mqc_scf_integral), intent(in) integral,
    character(len=*), optional orbString,
    integer(kind=int64), dimension(:), optional alphaOrbsIn,
    integer(kind=int64), dimension(:), optional betaOrbsIn,
    integer(kind=int64), intent(in), optional axis )
```

5.2.1.43 mqc_integral_scalar_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_scalar_multiply (  
    type(mqc_scf_integral), intent(in) integral,  
    type(mqc_scalar), intent(in) scalar )
```

5.2.1.44 mqc_integral_set_energy_list()

```
subroutine mqc_est::mqc_integral_set_energy_list (  
    class(mqc_scf_integral) integral,  
    integer(kind=int64), dimension(:), allocatable elist )
```

5.2.1.45 mqc_integral_sum()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_sum (  
    type(mqc_scf_integral), intent(in) integralA,  
    type(mqc_scf_integral), intent(in) integralB )
```

5.2.1.46 mqc_integral_swap_orbitals()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_swap_orbitals (  
    class(mqc_scf_integral), intent(in) integral,  
    integer(kind=int64), dimension(2), optional alphaOrbsIn,  
    integer(kind=int64), dimension(2), optional betaOrbsIn,  
    integer(kind=int64), intent(in), optional axis )
```

5.2.1.47 mqc_integral_transpose()

```
type(mqc_scf_integral) function mqc_est::mqc_integral_transpose (  
    type(mqc_scf_integral), intent(in) integral,  
    character(len=*), intent(in), optional label )
```

5.2.1.48 mqc_matrix_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_matrix_integral_multiply (
    type(mqc_matrix), intent(in) matrixA,
    type(mqc_scf_integral), intent(in) integralB,
    character(len=*), intent(in), optional label )
```

5.2.1.49 mqc_matrix_spinblockghf()

```
subroutine mqc_est::mqc_matrix_spinblockghf (
    class(*), intent(inout) array,
    integer(kind=int64), optional nelecs,
    integer(kind=int64), optional multi,
    integer(kind=int64), dimension(:), optional, allocatable elists )
```

5.2.1.50 mqc_matrix_undospinblockghf_eigenvalues()

```
subroutine mqc_est::mqc_matrix_undospinblockghf_eigenvalues (
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesIn,
    type(mqc_vector), intent(out) vectorOut )
```

5.2.1.51 mqc_matrix_undospinblockghf_integral()

```
subroutine mqc_est::mqc_matrix_undospinblockghf_integral (
    type(mqc_scf_integral), intent(in) integralIn,
    type(mqc_matrix), intent(out) matrixOut )
```

5.2.1.52 mqc_print_eigenvalues()

```
subroutine mqc_est::mqc_print_eigenvalues (
    class(mqc_scf_eigenvalues) eigenvalues,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

5.2.1.53 mqc_print_integral()

```
subroutine mqc_est::mqc_print_integral (
    class(mqc_scf_integral) integral,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

5.2.1.54 mqc_print_twoeris()

```
subroutine mqc_est::mqc_print_twoeris (
    class(mqc_twoeris) twoERIs,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

5.2.1.55 mqc_print_wavefunction()

```
subroutine mqc_est::mqc_print_wavefunction (
    class(mqc_wavefunction) wavefunction,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in), optional label )
```

5.2.1.56 mqc_scalar_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::mqc_scalar_integral_multiply (
    type(mqc_scalar), intent(in) scalar,
    type(mqc_scf_integral), intent(in) integral )
```

5.2.1.57 mqc_scf_eigenvalues_power()

```
subroutine mqc_est::mqc_scf_eigenvalues_power (
    class(mqc_scf_eigenvalues), intent(inout) eigenvalues,
    class(*) power )
```


5.2.1.58 mqc_scf_integral_contraction()

```
type(mqc_scalar) function mqc_est::mqc_scf_integral_contraction (
    type(mqc_scf_integral), intent(in) integral1,
    type(mqc_scf_integral), intent(in) integral2 )
```

5.2.1.59 mqc_scf_integral_determinant()

```
type(mqc_scalar) function mqc_est::mqc_scf_integral_determinant (
    class(mqc_scf_integral), intent(in) integral )
```

5.2.1.60 mqc_scf_integral_diagonalize()

```
subroutine mqc_est::mqc_scf_integral_diagonalize (
    class(mqc_scf_integral), intent(in) integral,
    type(mqc_scf_eigenvalues), intent(inout), optional eVals,
    type(mqc_scf_integral), intent(inout), optional eVecs )
```

5.2.1.61 mqc_scf_integral_generalized_eigensystem()

```
subroutine mqc_est::mqc_scf_integral_generalized_eigensystem (
    class(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), optional integralB,
    type(mqc_scf_eigenvalues), intent(inout), optional eVals,
    type(mqc_scf_integral), intent(inout), optional rEVecs,
    type(mqc_scf_integral), intent(inout), optional lEVecs )
```

5.2.1.62 mqc_scf_integral_inverse()

```
type(mqc_scf_integral) function mqc_est::mqc_scf_integral_inverse (
    class(mqc_scf_integral), intent(in) integral )
```

5.2.1.63 mqc_scf_integral_trace()

```
type(mqc_scalar) function mqc_est::mqc_scf_integral_trace (
    class(mqc_scf_integral), intent(in) integral )
```

5.2.1.64 `mqc_scf_transformation_matrix()`

```
subroutine mqc_est::mqc_scf_transformation_matrix (
    type(mqc_scf_integral), intent(in) overlap,
    type(mqc_scf_integral), intent(out) transform_matrix,
    integer(kind=int64), intent(out), optional nBasUse )
```

5.2.1.65 `mqc_twoeris_allocate()`

```
subroutine mqc_est::mqc_twoeris_allocate (
    class(mqc_twoeris) twoERIs,
    character(len=*) storageType,
    character(len=*) integralType,
    type(mqc_r4tensor), optional alpha,
    type(mqc_r4tensor), optional beta,
    type(mqc_r4tensor), optional alphaBeta,
    type(mqc_r4tensor), optional betaAlpha )
```

5.2.1.66 `mqc_twoeris_at()`

```
type(mqc_scalar) function mqc_est::mqc_twoeris_at (
    class(mqc_twoeris) twoERIs,
    integer(kind=int64), intent(in) i,
    integer(kind=int64), intent(in) j,
    integer(kind=int64), intent(in) k,
    integer(kind=int64), intent(in) l,
    character(len=64), optional spinBlock )
```

5.2.1.67 `slater_condon()`

```
type(mqc_scalar) function mqc_est::slater_condon (
    integer(kind=int64), intent(in) IOut,
    integer(kind=int64), intent(in) IPrint,
    type(mqc_scalar), intent(in) NBasisIn,
    type(mqc_determinant), intent(in) Determinants,
    integer(kind=int64), intent(in) L_A_String,
    integer(kind=int64), intent(in) L_B_String,
    integer(kind=int64), intent(in) R_A_String,
    integer(kind=int64), intent(in) R_B_String,
    type(mqc_scf_integral), intent(in) Core_Hamiltonian,
    type(mqc_twoeris), intent(in) ERIs,
    logical, intent(in) UHF )
```

5.2.1.68 twoeri_trans()

```
subroutine mqc_est::twoeri_trans (
    integer(kind=int64) IOut,
    integer(kind=int64) IPrint,
    type(mqc_scf_integral), intent(in) MO_Coeff,
    type(mqc_twoeris), intent(in) ERIs,
    type(mqc_twoeris), intent(out) MO_ERIs,
    logical UHF )
```


Chapter 6

Data Type Documentation

6.1 mqc_algebra::abs Interface Reference

Public Member Functions

- type(mqc_scalar) function mqc_scalar_get_abs_value (Scalar)
MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable
- type(mqc_vector) function mqc_vector_abs (A)

6.1.1 Member Function/Subroutine Documentation

6.1.1.1 mqc_scalar_get_abs_value()

```
type(mqc_scalar) function mqc_algebra::abs::mqc_scalar_get_abs_value (  
    class(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable

Purpose:

MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable.

Parameters

in	Scalar	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested
----	--------	--

Author

A. Mahler

Date

2018

6.1.1.2 mqc_vector_abs()

```
type(mqc_vector) function mqc_algebra::abs::mqc_vector_abs (
    class(mqc_vector), intent(in) A )
```

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

- src/mqc_algebra.F03

6.2 mqc_algebra::acos Interface Reference**Public Member Functions**

- type(mqc_scalar) function [mqc_scalar_acos](#) (Scalar)
MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar

6.2.1 Member Function/Subroutine Documentation**6.2.1.1 mqc_scalar_acos()**

```
type(mqc_scalar) function mqc_algebra::acos::mqc_scalar_acos (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar

Purpose:

MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.3 mqc_algebra::aimag Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_complex_imagpart` (ScalarIn)
MQC_Scalar_Complex_ImagPart is a function that returns the inaginary part of an MQC_Scalar
- `type(mqc_vector)` function `mqc_vector_complex_imagpart` (A)

6.3.1 Member Function/Subroutine Documentation

6.3.1.1 mqc_scalar_complex_imagpart()

```
type(mqc_scalar) function mqc_algebra::aimag::mqc_scalar_complex_imagpart (
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_ImagPart is a function that returns the inaginary part of an MQC_Scalar

Purpose:

MQC_Scalar_Complex_RealPart is a function that returns the imaginary part of an MQC_Scalar.

Parameters

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable
----	------------------------------	---

Author

L. M. Thompson

Date

2019

6.3.1.2 mqc_vector_complex_imagpart()

```
type(mqc_vector) function mqc_algebra::aimag::mqc_vector_complex_imagpart (
    class(mqc_vector), intent(in) A )
```

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

- src/[mqc_algebra.F03](#)

6.4 mqc_algebra::asin Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_asin](#) (Scalar)
MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar

6.4.1 Member Function/Subroutine Documentation

6.4.1.1 mqc_scalar_asin()

```
type(mqc_scalar) function mqc_algebra::asin::mqc_scalar_asin (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar

Purpose:

MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar.

Parameters

in	Scalar	Scalar is Type(MQC_Scalar) The argument of the function
----	--------	--

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.5 mqc_algebra::assignment(=) Interface Reference

Public Member Functions

- subroutine [mqc_input_integer_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an MQC_Scalar
- subroutine [mqc_input_real_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar
- subroutine [mqc_input_complex_scalar](#) (ScalarOut, ScalarIn)
MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an MQC_Scalar
- subroutine [mqc_output_mqcscalar_scalar](#) (ScalarOut, ScalarIn)
MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar
- subroutine [mqc_output_integer_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar
- subroutine [mqc_output_real_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar
- subroutine [mqc_output_complex_scalar](#) (ScalarOut, ScalarIn)
MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar
- subroutine [mqc_set_vector2vector](#) (VectorOut, VectorIn)
- subroutine [mqc_set_vector2integerarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_vector2realarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_vector2complexarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_matrix2matrix](#) (MatrixOut, MatrixIn)
- subroutine [mqc_set_matrix2integerarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2realarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2complexarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_integerarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_realarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_complexarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_array2tensor](#) (TensorOut, ArrayIn)

6.5.1 Member Function/Subroutine Documentation

6.5.1.1 `mqc_input_complex_scalar()`

```
subroutine mqc_algebra::assignment(=)::mqc_input_complex_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    complex(kind=real64), intent(in) ScalarIn )
```

MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an **MQC_Scalar**

Purpose:

`MQC_Input_Complex_Scalar` is a subroutine is used to set an intrinsic complex to an `MQC_Scalar`.

Parameters

<code>in, out</code>	<i>ScalarOut</i>	<p><code>ScalarOut</code> is <code>Type(MQC_Scalar)</code> The name of the output variable</p>
<code>in</code>	<i>ScalarIn</i>	<p><code>ScalarIn</code> is <code>Complex(kind=real64)</code> The value of the input variable</p>

Author

L. M. Thompson

Date

2017

6.5.1.2 `mqc_input_integer_scalar()`

```
subroutine mqc_algebra::assignment(=)::mqc_input_integer_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    integer(kind=int64), intent(in) ScalarIn )
```

MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an **MQC_Scalar**

Purpose:

`MQC_Input_Integer_Scalar` is a subroutine is used to set an intrinsic integer to an `MQC_Scalar`.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Integer(kind=int64) The value of the input variable

Author

L. M. Thompson

Date

2016

6.5.1.3 mqc_input_real_scalar()

```
subroutine mqc_algebra::assignment(=)::mqc_input_real_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    real(kind=real64), intent(in) ScalarIn )
```

MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar

Purpose:

MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Real(kind=real64) The value of the input variable

Author

L. M. Thompson

Date

2016

6.5.1.4 mqc_output_complex_scalar()

```
subroutine mqc_algebra::assignment(=)::mqc_output_complex_scalar (
    complex(kind=real64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar

Purpose:

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Complex(kind=real64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2017

6.5.1.5 mqc_output_integer_scalar()

```
subroutine mqc_algebra::assignment(=)::mqc_output_integer_scalar (
    integer(kind=int64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar

Purpose:

MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Integer(kind=int64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

6.5.1.6 mqc_output_mqcscalar_scalar()

```
subroutine mqc_algebra::assignment(=)::mqc_output_mqcscalar_scalar (
    type(mqc_scalar), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar

Purpose:

MQC_Output_MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Type(MQC_Scalar) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

6.5.1.7 mqc_output_real_scalar()

```
subroutine mqc_algebra::assignment(=)::mqc_output_real_scalar (
    real(kind=real64), intent(inout) ScalarOut,
    type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar

Purpose:

MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar.

Parameters

in, out	<i>ScalarOut</i>	ScalarOut is Real(kind=real64) The name of the output variable
in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The value of the input variable

Author

L. M. Thompson

Date

2016

6.5.1.8 mqc_set_array2tensor()

```
subroutine mqc_algebra::assignment(=)::mqc_set_array2tensor (
    type(mqc_r4tensor), intent(inout) TensorOut,
    class(*), dimension(:, :, :, :), intent(in) ArrayIn )
```

6.5.1.9 mqc_set_array2vector_complex()

```
subroutine mqc_algebra::assignment(=)::mqc_set_array2vector_complex (
    type(mqc_vector), intent(inout) VectorOut,
    complex(kind=real64), dimension(:), intent(in) ArrayIn )
```

6.5.1.10 mqc_set_array2vector_integer()

```
subroutine mqc_algebra::assignment(=)::mqc_set_array2vector_integer (
    type(mqc_vector), intent(inout) VectorOut,
    integer(kind=int64), dimension(:), intent(in) ArrayIn )
```

6.5.1.11 mqc_set_array2vector_real()

```
subroutine mqc_algebra::assignment(=)::mqc_set_array2vector_real (
    type(mqc_vector), intent(inout) VectorOut,
    real(kind=real64), dimension(:), intent(in) ArrayIn )
```

6.5.1.12 mqc_set_complexarray2matrix()

```
subroutine mqc_algebra::assignment(=)::mqc_set_complexarray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    complex(kind=real64), dimension(:,:), intent(in) ArrayIn )
```

6.5.1.13 mqc_set_integerarray2matrix()

```
subroutine mqc_algebra::assignment(=)::mqc_set_integerarray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    integer(kind=int64), dimension(:,:), intent(in) ArrayIn )
```

6.5.1.14 mqc_set_matrix2complexarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_matrix2complexarray (
    complex(kind=real64), dimension(:,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

6.5.1.15 mqc_set_matrix2integerarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_matrix2integerarray (
    integer(kind=int64), dimension(:,,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

6.5.1.16 mqc_set_matrix2matrix()

```
subroutine mqc_algebra::assignment(=)::mqc_set_matrix2matrix (
    class(mqc_matrix), intent(inout) MatrixOut,
    class(mqc_matrix), intent(in) MatrixIn )
```

6.5.1.17 mqc_set_matrix2realarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_matrix2realarray (
    real(kind=real64), dimension(:,,:), intent(inout), allocatable ArrayOut,
    type(mqc_matrix), intent(in) MatrixIn )
```

6.5.1.18 mqc_set_realarray2matrix()

```
subroutine mqc_algebra::assignment(=)::mqc_set_realarray2matrix (
    type(mqc_matrix), intent(inout) MatrixOut,
    real(kind=real64), dimension(:,,:), intent(in) ArrayIn )
```

6.5.1.19 mqc_set_vector2complexarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_vector2complexarray (
    complex(kind=real64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```

6.5.1.20 mqc_set_vector2integerarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_vector2integerarray (
    integer(kind=int64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```


6.5.1.21 mqc_set_vector2realarray()

```
subroutine mqc_algebra::assignment(=)::mqc_set_vector2realarray (
    real(kind=real64), dimension(:), intent(inout), allocatable ArrayOut,
    type(mqc_vector), intent(in) VectorIn )
```

6.5.1.22 mqc_set_vector2vector()

```
subroutine mqc_algebra::assignment(=)::mqc_set_vector2vector (
    class(mqc_vector), intent(inout) VectorOut,
    class(mqc_vector), intent(in) VectorIn )
```

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

- src/[mqc_algebra.F03](#)

6.6 mqc_est::assignment(=) Interface Reference

Public Member Functions

- subroutine [mqc_integral_output_array](#) (matrixOut, integralIn)
- subroutine [mqc_eigenvalues_output_array](#) (vectorOut, eigenvaluesIn)

6.6.1 Member Function/Subroutine Documentation

6.6.1.1 mqc_eigenvalues_output_array()

```
subroutine mqc_est::assignment(=)::mqc_eigenvalues_output_array (
    type(mqc_vector), intent(inout) vectorOut,
    class(mqc_scf_eigenvalues), intent(in) eigenvaluesIn )
```

6.6.1.2 mqc_integral_output_array()

```
subroutine mqc_est::assignment(=)::mqc_integral_output_array (
    type(mqc_matrix), intent(inout) matrixOut,
    class(mqc_scf_integral), intent(in) integralIn )
```

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

- src/[mqc_est.F03](#)

6.7 mqc_algebra::atan Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_atan` (Scalar)
MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar

6.7.1 Member Function/Subroutine Documentation

6.7.1.1 mqc_scalar_atan()

```
type(mqc_scalar) function mqc_algebra::atan::mqc_scalar_atan (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar

Purpose:

MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar.

Parameters

in	Scalar	Scalar is Type(MQC_Scalar) The argument of the function
----	--------	--

Author

L. M. Thompson

Date

2019

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

- `src/mqc_algebra.F03`

6.8 mqc_algebra::atan2 Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_atan2` (Scalar)
MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram

6.8.1 Member Function/Subroutine Documentation

6.8.1.1 mqc_scalar_atan2()

```
type(mqc_scalar) function mqc_algebra::atan2::mqc_scalar_atan2 (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram

Purpose:

MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram.

Parameters

in	Scalar	Scalar is Type(MQC_Scalar) The argument of the function
----	--------	--

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.9 mqc_algebra::cmplx Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_cmplx](#) (Scalar1, Scalar2)
MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars
- type([mqc_vector](#)) function [mqc_vector_cmplx](#) (Vector1, Vector2)

6.9.1 Member Function/Subroutine Documentation

6.9.1.1 `mqc_scalar_cmplx()`

```
type(mqc_scalar) function mqc_algebra::cmplx::mqc_scalar_cmplx (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars

Purpose:

MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_Scalar variables.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The real part of MQC_Scalar_Cmplx
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The imaginary part of MQC_Scalar_Cmplx

Author

L. M. Thompson

Date

2019

6.9.1.2 `mqc_vector_cmplx()`

```
type(mqc_vector) function mqc_algebra::cmplx::mqc_vector_cmplx (
    type(mqc_vector), intent(in) Vector1,
    type(mqc_vector), intent(in) Vector2 )
```

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

- [src/mqc_algebra.F03](#)

6.10 `mqc_algebra::conjg` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_complex_conjugate` (`ScalarIn`)
`MQC_Scalar_Complex_Conjugate` is a function that returns the complex conjugate of an `MQC_Scalar`

6.10.1 Member Function/Subroutine Documentation

6.10.1.1 `mqc_scalar_complex_conjugate()`

```
type(mqc_scalar) function mqc_algebra::conjg::mqc_scalar_complex_conjugate (
    type(mqc_scalar), intent(in) ScalarIn )
```

`MQC_Scalar_Complex_Conjugate` is a function that returns the complex conjugate of an `MQC_Scalar`

Purpose:

`MQC_Scalar_Complex_Conjugate` is a function that returns the complex conjugate of an `MQC_Scalar`.

Parameters

<code>in</code>	<i><code>ScalarIn</code></i>	<p><code>ScalarIn</code> is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> input variable</p>
-----------------	------------------------------	---

Author

L. M. Thompson

Date

2018

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

- `src/mqc_algebra.F03`

6.11 `mqc_algebra::contraction` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_matrix_matrix_contraction` (`Matrix1`, `Matrix2`)

6.11.1 Member Function/Subroutine Documentation

6.11.1.1 `mqc_matrix_matrix_contraction()`

```
type(mqc_scalar) function mqc_algebra::contraction::mqc_matrix_matrix_contraction (
    type(mqc_matrix), intent(in) Matrix1,
    type(mqc_matrix), intent(in) Matrix2 )
```

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

- [src/mqc_algebra.F03](#)

6.12 `mqc_est::contraction` Interface Reference

Public Member Functions

- type(mqc_scalar) function [mqc_scf_integral_contraction](#) (integral1, integral2)
- type([mqc_scf_integral](#)) function [mqc_eri_integral_contraction](#) (eris, integral, label)

6.12.1 Member Function/Subroutine Documentation

6.12.1.1 `mqc_eri_integral_contraction()`

```
type(mqc_scf_integral) function mqc_est::contraction::mqc_eri_integral_contraction (
    type(mqc_twoeris), intent(in) eris,
    type(mqc_scf_integral), intent(in) integral,
    character(len=*), optional label )
```

6.12.1.2 `mqc_scf_integral_contraction()`

```
type(mqc_scalar) function mqc_est::contraction::mqc_scf_integral_contraction (
    type(mqc_scf_integral), intent(in) integral1,
    type(mqc_scf_integral), intent(in) integral2 )
```

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

- [src/mqc_est.F03](#)

6.13 mqc_algebra::cos Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_cos](#) (Scalar)
MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar

6.13.1 Member Function/Subroutine Documentation

6.13.1.1 mqc_scalar_cos()

```
type(mqc\_scalar) function mqc_algebra::cos::mqc_scalar_cos (
    type(mqc\_scalar), intent(in) Scalar )
```

MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar

Purpose:

MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The argument of the function
----	---------------	--

Author

L. M. Thompson

Date

2019

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

- src/[mqc_algebra.F03](#)

6.14 mqc_algebra::dagger Interface Reference

Public Member Functions

- type([mqc_vector](#)) function [mqc_vector_conjugate_transpose](#) (Vector)
- type([mqc_matrix](#)) function [mqc_matrix_conjugate_transpose](#) (Matrix)

6.14.1 Member Function/Subroutine Documentation

6.14.1.1 `mqc_matrix_conjugate_transpose()`

```
type(mqc\_matrix) function mqc_algebra::dagger::mqc_matrix_conjugate_transpose (
    class(mqc\_matrix), intent(in) Matrix )
```

6.14.1.2 `mqc_vector_conjugate_transpose()`

```
type(mqc\_vector) function mqc_algebra::dagger::mqc_vector_conjugate_transpose (
    class(mqc\_vector), intent(in) Vector )
```

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

- [src/mqc_algebra.F03](#)

6.15 `mqc_est::dagger` Interface Reference

Public Member Functions

- type([mqc_scf_integral](#)) function [mqc_integral_conjugate_transpose](#) (integral, label)

6.15.1 Member Function/Subroutine Documentation

6.15.1.1 `mqc_integral_conjugate_transpose()`

```
type(mqc\_scf\_integral) function mqc_est::dagger::mqc_integral_conjugate_transpose (
    type(mqc\_scf\_integral), intent(in) integral,
    character(len=*), intent(in), optional label )
```

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

- [src/mqc_est.F03](#)

6.16 `mqc_algebra::dot_product` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_vectorvectordotproduct` (Vector1, Vector2)

6.16.1 Member Function/Subroutine Documentation

6.16.1.1 `mqc_vectorvectordotproduct()`

```
type(mqc_scalar) function mqc_algebra::dot_product::mqc_vectorvectordotproduct (
    type(mqc_vector), intent(in) Vector1,
    type(mqc_vector), intent(in) Vector2 )
```

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

- `src/mqc_algebra.F03`

6.17 `mqc_est::dot_product` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_eigenvalue_eigenvalue_dotproduct` (eigenvalueA, eigenvalueB)

6.17.1 Member Function/Subroutine Documentation

6.17.1.1 `mqc_eigenvalue_eigenvalue_dotproduct()`

```
type(mqc_scalar) function mqc_est::dot_product::mqc_eigenvalue_eigenvalue_dotproduct (
    type(mqc_scf_eigenvalues), intent(in) eigenvalueA,
    type(mqc_scf_eigenvalues), intent(in) eigenvalueB )
```

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

- `src/mqc_est.F03`

6.18 mqc_algebra::matmul Interface Reference

Public Member Functions

- type([mqc_matrix](#)) function [mqc_matrixmatrixdotproduct](#) (MA, MB)
- type([mqc_vector](#)) function [mqc_matrixvectordotproduct](#) (MA, VB)
- type([mqc_vector](#)) function [mqc_vectormatrixdotproduct](#) (VA, MB)

6.18.1 Member Function/Subroutine Documentation

6.18.1.1 mqc_matrixmatrixdotproduct()

```
type(mqc\_matrix) function mqc_algebra::matmul::mqc_matrixmatrixdotproduct (
    type(mqc\_matrix), intent(in) MA,
    type(mqc\_matrix), intent(in) MB )
```

6.18.1.2 mqc_matrixvectordotproduct()

```
type(mqc\_vector) function mqc_algebra::matmul::mqc_matrixvectordotproduct (
    type(mqc\_matrix), intent(in) MA,
    type(mqc\_vector), intent(in) VB )
```

6.18.1.3 mqc_vectormatrixdotproduct()

```
type(mqc\_vector) function mqc_algebra::matmul::mqc_vectormatrixdotproduct (
    type(mqc\_vector), intent(in) VA,
    type(mqc\_matrix), intent(in) MB )
```

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

- [src/mqc_algebra.F03](#)

6.19 mqc_est::matmul Interface Reference

Public Member Functions

- type([mqc_scf_integral](#)) function [mqc_integral_matrix_multiply](#) (integralA, matrixB, label)
- type([mqc_scf_integral](#)) function [mqc_matrix_integral_multiply](#) (matrixA, integralB, label)
- type([mqc_scf_integral](#)) function [mqc_integral_integral_multiply](#) (integralA, integralB, label)
- type([mqc_scf_integral](#)) function [mqc_integral_eigenvalues_multiply](#) (integralA, eigenvaluesB, label)
- type([mqc_scf_integral](#)) function [mqc_eigenvalues_integral_multiply](#) (eigenvaluesA, integralB, label)
- type([mqc_scf_eigenvalues](#)) function [mqc_eigenvalues_eigenvalues_multiply](#) (eigenvaluesA, eigenvaluesB, label)

6.19.1 Member Function/Subroutine Documentation

6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()

```
type(mqc\_scf\_eigenvalues) function mqc_est::matmul::mqc_eigenvalues_eigenvalues_multiply (
    type(mqc\_scf\_eigenvalues), intent(in) eigenvaluesA,
    type(mqc\_scf\_eigenvalues), intent(in) eigenvaluesB,
    character(len=*), intent(in), optional label )
```

6.19.1.2 mqc_eigenvalues_integral_multiply()

```
type(mqc\_scf\_integral) function mqc_est::matmul::mqc_eigenvalues_integral_multiply (
    type(mqc\_scf\_eigenvalues), intent(in) eigenvaluesA,
    type(mqc\_scf\_integral), intent(in) integralB,
    character(len=*), intent(in), optional label )
```

6.19.1.3 mqc_integral_eigenvalues_multiply()

```
type(mqc\_scf\_integral) function mqc_est::matmul::mqc_integral_eigenvalues_multiply (
    type(mqc\_scf\_integral), intent(in) integralA,
    type(mqc\_scf\_eigenvalues), intent(in) eigenvaluesB,
    character(len=*), intent(in), optional label )
```

6.19.1.4 mqc_integral_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::matmul::mqc_integral_integral_multiply (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), intent(in) integralB,
    character(len=*), intent(in), optional label )
```

6.19.1.5 mqc_integral_matrix_multiply()

```
type(mqc_scf_integral) function mqc_est::matmul::mqc_integral_matrix_multiply (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_matrix), intent(in) matrixB,
    character(len=*), intent(in), optional label )
```

6.19.1.6 mqc_matrix_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::matmul::mqc_matrix_integral_multiply (
    type(mqc_matrix), intent(in) matrixA,
    type(mqc_scf_integral), intent(in) integralB,
    character(len=*), intent(in), optional label )
```

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

- [src/mqc_est.F03](#)

6.20 mqc_algebra::matrix_symm2sq Interface Reference

Public Member Functions

- subroutine [matrix_symm2sq_integer](#) (N, I_Symm, I_Sq)
- subroutine [matrix_symm2sq_real](#) (N, A_Symm, A_Sq)
- subroutine [matrix_symm2sq_complex](#) (N, A_Symm, A_Sq)

6.20.1 Member Function/Subroutine Documentation

6.20.1.1 matrix_symm2sq_complex()

```
subroutine mqc_algebra::matrix_symm2sq::matrix_symm2sq_complex (
    integer(kind=int64), intent(in) N,
    complex(kind=real64), dimension(:), intent(in) A_Symm,
    complex(kind=real64), dimension(n,n), intent(out) A_Sq )
```

6.20.1.2 matrix_symm2sq_integer()

```
subroutine mqc_algebra::matrix_symm2sq::matrix_symm2sq_integer (
    integer(kind=int64), intent(in) N,
    integer(kind=int64), dimension(:), intent(in) I_Symm,
    integer(kind=int64), dimension(n,n), intent(out) I_Sq )
```

6.20.1.3 matrix_symm2sq_real()

```
subroutine mqc_algebra::matrix_symm2sq::matrix_symm2sq_real (
    integer(kind=int64), intent(in) N,
    real(kind=real64), dimension(:), intent(in) A_Symm,
    real(kind=real64), dimension(n,n), intent(out) A_Sq )
```

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

- [src/mqc_algebra.F03](#)

6.21 mqc_algebra::mqc_cast_complex Interface Reference

Public Member Functions

- type([mqc_vector](#)) function [mqc_vector_cast_complex](#) (VA)
- type([mqc_matrix](#)) function [mqc_matrix_cast_complex](#) (MA)

6.21.1 Member Function/Subroutine Documentation

6.21.1.1 mqc_matrix_cast_complex()

```
type(mqc\_matrix) function mqc_algebra::mqc_cast_complex::mqc_matrix_cast_complex (
    type(mqc\_matrix), intent(in) MA )
```

6.21.1.2 `mqc_vector_cast_complex()`

```
type(mqc\_vector) function mqc_algebra::mqc_cast_complex::mqc_vector_cast_complex (
    type(mqc\_vector), intent(in) VA )
```

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

- [src/mqc_algebra.F03](#)

6.22 `mqc_algebra::mqc_cast_real` Interface Reference

Public Member Functions

- type([mqc_vector](#)) function [mqc_vector_cast_real](#) (VA)
- type([mqc_matrix](#)) function [mqc_matrix_cast_real](#) (MA)

6.22.1 Member Function/Subroutine Documentation

6.22.1.1 `mqc_matrix_cast_real()`

```
type(mqc\_matrix) function mqc_algebra::mqc_cast_real::mqc_matrix_cast_real (
    type(mqc\_matrix), intent(in) MA )
```

6.22.1.2 `mqc_vector_cast_real()`

```
type(mqc\_vector) function mqc_algebra::mqc_cast_real::mqc_vector_cast_real (
    type(mqc\_vector), intent(in) VA )
```

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

- [src/mqc_algebra.F03](#)

6.23 `mqc_est::mqc_determinant` Type Reference

Public Attributes

- type([mqc_determinant_string](#)) [strings](#)
- character(len=64) [order](#)
- integer(kind=int64) [ndets](#)
- integer(kind=int64) [nalpstr](#)
- integer(kind=int64) [nbetstr](#)

6.23.1 Member Data Documentation

6.23.1.1 nalpstr

```
integer(kind=int64) mqc_est::mqc_determinant::nalpstr
```

6.23.1.2 nbetstr

```
integer(kind=int64) mqc_est::mqc_determinant::nbetstr
```

6.23.1.3 ndets

```
integer(kind=int64) mqc_est::mqc_determinant::ndets
```

6.23.1.4 order

```
character(len=64) mqc_est::mqc_determinant::order
```

6.23.1.5 strings

```
type(mqc_determinant_string) mqc_est::mqc_determinant::strings
```

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

- [src/mqc_est.F03](#)

6.24 mqc_est::mqc_determinant_string Type Reference

Public Attributes

- type(mqc_matrix) [alpha](#)
- type(mqc_matrix) [beta](#)

6.24.1 Member Data Documentation

6.24.1.1 alpha

```
type(mqc_matrix) mqc_est::mqc_determinant_string::alpha
```

6.24.1.2 beta

```
type(mqc_matrix) mqc_est::mqc_determinant_string::beta
```

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

- src/[mqc_est.F03](#)

6.25 mqc_algebra::mqc_have_complex Interface Reference

Public Member Functions

- logical function [mqc_vector_havecomplex](#) (Vector)
- logical function [mqc_matrix_havecomplex](#) (Matrix)

6.25.1 Member Function/Subroutine Documentation

6.25.1.1 mqc_matrix_havecomplex()

```
logical function mqc_algebra::mqc_have_complex::mqc_matrix_havecomplex (
    type(mqc\_matrix), intent(in) Matrix )
```

6.25.1.2 mqc_vector_havecomplex()

```
logical function mqc_algebra::mqc_have_complex::mqc_vector_havecomplex (
    type(mqc\_vector), intent(in) Vector )
```

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

- src/[mqc_algebra.F03](#)

6.26 `mqc_algebra::mqc_have_int` Interface Reference

Public Member Functions

- logical function [mqc_vector_haveinteger](#) (Vector)
- logical function [mqc_matrix_haveinteger](#) (Matrix)

6.26.1 Member Function/Subroutine Documentation

6.26.1.1 `mqc_matrix_haveinteger()`

```
logical function mqc_algebra::mqc_have_int::mqc_matrix_haveinteger (  
    type(mqc\_matrix), intent(in) Matrix )
```

6.26.1.2 `mqc_vector_haveinteger()`

```
logical function mqc_algebra::mqc_have_int::mqc_vector_haveinteger (  
    type(mqc\_vector), intent(in) Vector )
```

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

- src/[mqc_algebra.F03](#)

6.27 `mqc_algebra::mqc_have_real` Interface Reference

Public Member Functions

- logical function [mqc_vector_havereal](#) (Vector)
- logical function [mqc_matrix_havereal](#) (Matrix)

6.27.1 Member Function/Subroutine Documentation

6.27.1.1 mqc_matrix_havereal()

```
logical function mqc_algebra::mqc_have_real::mqc_matrix_havereal (
    type(mqc_matrix), intent(in) Matrix )
```

6.27.1.2 mqc_vector_havereal()

```
logical function mqc_algebra::mqc_have_real::mqc_vector_havereal (
    type(mqc_vector), intent(in) Vector )
```

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

- src/mqc_algebra.F03

6.28 mqc_algebra::mqc_matrix Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_matrix_algebra1](#)
- Procedure, public [initialize](#) => [mqc_matrix_initialize](#)
- Procedure, public [init](#) => [mqc_matrix_initialize](#)
- Procedure, public [identity](#) => [mqc_matrix_identity](#)
- Procedure, public [set](#) => [mqc_matrix_set](#)
- Procedure, public [norm](#) => [mqc_matrix_norm](#)
- Procedure, public [transpose](#) => [mqc_matrix_transpose](#)
- Procedure, public [dagger](#) => [mqc_matrix_conjugate_transpose](#)
- Procedure, public [diag](#) => [mqc_matrix_diagonalize](#)
- Procedure, public [svd](#) => [mqc_matrix_svd](#)
- Procedure, public [eigensys](#) => [mqc_matrix_generalized_eigensystem](#)
- Procedure, public [inv](#) => [mqc_matrix_inverse](#)
- Procedure, public [det](#) => [mqc_matrix_determinant](#)
- Procedure, public [trace](#) => [mqc_matrix_trace](#)
- Procedure, public [rmsmax](#) => [mqc_matrix_rms_max](#)
- Procedure, public [sqrt](#) => [mqc_matrix_sqrt](#)
- Procedure, public [at](#) => [mqc_matrix_scalar_at](#)
- Procedure, public [vat](#) => [mqc_matrix_vector_at](#)
- Procedure, public [mat](#) => [mqc_matrix_matrix_at](#)
- Procedure, public [put](#) => [mqc_matrix_scalar_put](#)
- Procedure, public [vput](#) => [mqc_matrix_vector_put](#)
- Procedure, public [mput](#) => [mqc_matrix_matrix_put](#)
- Procedure, public [s_type](#) => [mqc_matrix_storagetype](#)

Public Attributes

- [real](#)(kind=real64), dimension(:,.), allocatable [matr](#)
- [integer](#)(kind=int64), dimension(:,.), allocatable [mati](#)
- [complex](#)(kind=real64), dimension(:,.), allocatable [matc](#)

6.28.1 Member Function/Subroutine Documentation

6.28.1.1 `at()`

Procedure, public mqc_algebra::mqc_matrix::at ()

6.28.1.2 `dagger()`

Procedure, public mqc_algebra::mqc_matrix::dagger ()

6.28.1.3 `det()`

Procedure, public mqc_algebra::mqc_matrix::det ()

6.28.1.4 `diag()`

Procedure, public mqc_algebra::mqc_matrix::diag ()

6.28.1.5 `eigensys()`

Procedure, public mqc_algebra::mqc_matrix::eigensys ()

6.28.1.6 identity()

Procedure, public mqc_algebra::mqc_matrix::identity ()

6.28.1.7 init()

Procedure, public mqc_algebra::mqc_matrix::init ()

6.28.1.8 initialize()

Procedure, public mqc_algebra::mqc_matrix::initialize ()

6.28.1.9 inv()

Procedure, public mqc_algebra::mqc_matrix::inv ()

6.28.1.10 mat()

Procedure, public mqc_algebra::mqc_matrix::mat ()

6.28.1.11 mput()

Procedure, public mqc_algebra::mqc_matrix::mput ()

6.28.1.12 norm()

Procedure, public mqc_algebra::mqc_matrix::norm ()

6.28.1.13 print()

Procedure, public mqc_algebra::mqc_matrix::print ()

6.28.1.14 put()

Procedure, public mqc_algebra::mqc_matrix::put ()

6.28.1.15 rmsmax()

Procedure, public mqc_algebra::mqc_matrix::rmsmax ()

6.28.1.16 s_type()

Procedure, public mqc_algebra::mqc_matrix::s_type ()

6.28.1.17 set()

Procedure, public mqc_algebra::mqc_matrix::set ()

6.28.1.18 sqrt()

Procedure, public mqc_algebra::mqc_matrix::sqrt ()

6.28.1.19 svd()

Procedure, public mqc_algebra::mqc_matrix::svd ()

6.28.1.20 trace()

```
Procedure, public mqc_algebra::mqc_matrix::trace ( )
```

6.28.1.21 transpose()

```
Procedure, public mqc_algebra::mqc_matrix::transpose ( )
```

6.28.1.22 vat()

```
Procedure, public mqc_algebra::mqc_matrix::vat ( )
```

6.28.1.23 vput()

```
Procedure, public mqc_algebra::mqc_matrix::vput ( )
```

6.28.2 Member Data Documentation**6.28.2.1 matc**

```
complex(kind=real64), dimension(:,:), allocatable mqc_algebra::mqc_matrix::matc
```

6.28.2.2 mati

```
integer(kind=int64), dimension(:,:), allocatable mqc_algebra::mqc_matrix::mati
```

6.28.2.3 matr

```
real(kind=real64), dimension(:,:), allocatable mqc_algebra::mqc_matrix::matr
```

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

- src/mqc_algebra.F03

6.29 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference

Public Member Functions

- subroutine [mqc_matrix_diagmatrix_put_integer](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_real](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_complex](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_vector](#) (diagVectorIn, mat)

6.29.1 Member Function/Subroutine Documentation

6.29.1.1 mqc_matrix_diagmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_complex (
    class(mqc_matrix), intent(inout) mat,
    complex(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

6.29.1.2 mqc_matrix_diagmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_integer (
    class(mqc_matrix), intent(inout) mat,
    integer(kind=int64), dimension(:), intent(in) diagMatrixIn )
```

6.29.1.3 mqc_matrix_diagmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_real (
    class(mqc_matrix), intent(inout) mat,
    real(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

6.29.1.4 mqc_matrix_diagmatrix_put_vector()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_vector (
    class(mqc_vector), intent(in) diagVectorIn,
    class(mqc_matrix), intent(inout) mat )
```

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

- src/[mqc_algebra.F03](#)

6.30 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference

Public Member Functions

- subroutine [mqc_matrix_symmmatrix_put_integer](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_real](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_complex](#) (mat, symmMatrixIn)

6.30.1 Member Function/Subroutine Documentation

6.30.1.1 mqc_matrix_symmmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_complex (
    class(mqc_matrix), intent(inout) mat,
    complex(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

6.30.1.2 mqc_matrix_symmmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_integer (
    class(mqc_matrix), intent(inout) mat,
    integer(kind=int64), dimension(:), intent(in) symmMatrixIn )
```

6.30.1.3 mqc_matrix_symmmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_real (
    class(mqc_matrix), intent(inout) mat,
    real(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

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

- src/[mqc_algebra.F03](#)

6.31 mqc_est::mqc_matrix_undospinblockghf Interface Reference

Public Member Functions

- subroutine [mqc_matrix_undospinblockghf_eigenvalues](#) (eigenvaluesIn, vectorOut)
- subroutine [mqc_matrix_undospinblockghf_integral](#) (integralIn, matrixOut)

6.31.1 Member Function/Subroutine Documentation

6.31.1.1 mqc_matrix_undospinblockghf_eigenvalues()

```
subroutine mqc_est::mqc_matrix_undospinblockghf::mqc_matrix_undospinblockghf_eigenvalues (
    type(mqc_scf_eigenvalues), intent(in) eigenvaluesIn,
    type(mqc_vector), intent(out) vectorOut )
```

6.31.1.2 mqc_matrix_undospinblockghf_integral()

```
subroutine mqc_est::mqc_matrix_undospinblockghf::mqc_matrix_undospinblockghf_integral (
    type(mqc_scf_integral), intent(in) integralIn,
    type(mqc_matrix), intent(out) matrixOut )
```

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

- [src/mqc_est.F03](#)

6.32 mqc_algebra::mqc_print Interface Reference

Public Member Functions

- subroutine [mqc_print_scalar_algebra1](#) (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)
MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar
- subroutine [mqc_print_vector_algebra1](#) (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_print_matrix_algebra1](#) (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_print_r4tensor_algebra1](#) (Tensor, IOut, Header, blank_at_top, blank_at_bottom)

6.32.1 Member Function/Subroutine Documentation

6.32.1.1 mqc_print_matrix_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_matrix_algebra1 (
    class(mqc_matrix), intent(in) Matrix,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )
```

6.32.1.2 mqc_print_r4tensor_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_r4tensor_algebra1 (
    class(mqc_r4tensor), intent(in) Tensor,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in), optional Header,
    logical, optional blank_at_top,
    logical, optional blank_at_bottom )
```

6.32.1.3 mqc_print_scalar_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_scalar_algebra1 (
    class(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )
```

MQC_Print_Scalar_Algebra1 is a subroutine used to print an **MQC_Scalar**

Purpose:

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar. Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank lines before or after output.

Parameters

in	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The variable to be printed
in	<i>IOut</i>	IOut is Integer(kind=int64) The Fortran file number to print to

Parameters

in	<i>Header</i>	Header is Character(Len=*) The title to print along with Scalar
in	<i>Blank_At_Top</i>	Blank_At_Top is Logical,Optional = .True.: print blank line above output = .False.: do not print blank line above output
in	<i>Blank_At_Bottom</i>	Blank_At_Bottom is Logical,Optional = .True.: print blank line below output = .False.: do not print blank line below output

Author

L. M. Thompson

Date

2016

6.32.1.4 mqc_print_vector_algebra1()

```

subroutine mqc_algebra::mqc_print::mqc_print_vector_algebra1 (
    class(mqc_vector), intent(in) Vector,
    integer(kind=int64), intent(in) IOut,
    character(len=*), intent(in) Header,
    logical, intent(in), optional Verbose,
    logical, intent(in), optional Blank_At_Top,
    logical, intent(in), optional Blank_At_Bottom )

```

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

- src/[mqc_algebra.F03](#)

6.33 mqc_est::mqc_print Interface Reference

Public Member Functions

- subroutine [mqc_print_wavefunction](#) (wavefunction, iOut, label)
- subroutine [mqc_print_integral](#) (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_eigenvalues](#) (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_twoeris](#) (twoERIs, iOut, header, blank_at_top, blank_at_bottom)

6.33.1 Member Function/Subroutine Documentation

6.33.1.1 `mqc_print_eigenvalues()`

```
subroutine mqc_est::mqc_print::mqc_print_eigenvalues (
    class(mqc_scf_eigenvalues) eigenvalues,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

6.33.1.2 `mqc_print_integral()`

```
subroutine mqc_est::mqc_print::mqc_print_integral (
    class(mqc_scf_integral) integral,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

6.33.1.3 `mqc_print_twoeris()`

```
subroutine mqc_est::mqc_print::mqc_print_twoeris (
    class(mqc_twoeris) twoERIs,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in) header,
    logical, intent(in), optional blank_at_top,
    logical, intent(in), optional blank_at_bottom )
```

6.33.1.4 `mqc_print_wavefunction()`

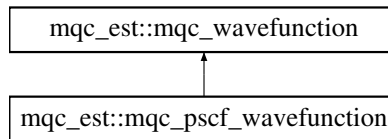
```
subroutine mqc_est::mqc_print::mqc_print_wavefunction (
    class(mqc_wavefunction) wavefunction,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in), optional label )
```

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

- [src/mqc_est.F03](#)

6.34 mqc_est::mqc_pscf_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_pscf_wavefunction:



Public Attributes

- integer(kind=int64) [ncore](#)
- integer(kind=int64) [nval](#)
- integer(kind=int64) [nactive](#)
- integer(kind=int64) [nfrz](#)
- type(mqc_matrix) [pscf_amplitudes](#)
- type(mqc_vector) [pscf_energies](#)

Additional Inherited Members

6.34.1 Member Data Documentation

6.34.1.1 nactive

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nactive
```

6.34.1.2 ncore

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::ncore
```

6.34.1.3 nfrz

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nfrz
```

6.34.1.4 nval

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nval
```

6.34.1.5 pscf_amplitudes

```
type(mqc_matrix) mqc_est::mqc_pscf_wavefunction::pscf_amplitudes
```

6.34.1.6 pscf_energies

```
type(mqc_vector) mqc_est::mqc_pscf_wavefunction::pscf_energies
```

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

- [src/mqc_est.F03](#)

6.35 mqc_algebra::mqc_r4tensor Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_r4tensor_algebra1](#)
- Procedure, public [at](#) => [mqc_r4tensor_at](#)
- Procedure, public [put](#) => [mqc_r4tensor_put](#)
- Procedure, public [initialize](#) => [mqc_r4tensor_initialize](#)
- Procedure, public [init](#) => [mqc_r4tensor_initialize](#)

6.35.1 Member Function/Subroutine Documentation

6.35.1.1 at()

```
Procedure, public mqc_algebra::mqc_r4tensor::at ( )
```

6.35.1.2 init()

Procedure, public mqc_algebra::mqc_r4tensor::init ()

6.35.1.3 initialize()

Procedure, public mqc_algebra::mqc_r4tensor::initialize ()

6.35.1.4 print()

Procedure, public mqc_algebra::mqc_r4tensor::print ()

6.35.1.5 put()

Procedure, public mqc_algebra::mqc_r4tensor::put ()

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

- src/[mqc_algebra.F03](#)

6.36 mqc_algebra::mqc_scalar Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_scalar_algebra1](#)
- Procedure, public [rval](#) => [mqc_scalar_get_intrinsic_real](#)
- Procedure, public [ival](#) => [mqc_scalar_get_intrinsic_integer](#)
- Procedure, public [cval](#) => [mqc_scalar_get_intrinsic_complex](#)
- Procedure, public [abs](#) => [mqc_scalar_get_abs_value](#)
- Procedure, public [random](#) => [mqc_scalar_get_random_value](#)

6.36.1 Member Function/Subroutine Documentation

6.36.1.1 abs()

Procedure, public mqc_algebra::mqc_scalar::abs ()

6.36.1.2 cval()

Procedure, public mqc_algebra::mqc_scalar::cval ()

6.36.1.3 ival()

Procedure, public mqc_algebra::mqc_scalar::ival ()

6.36.1.4 print()

Procedure, public mqc_algebra::mqc_scalar::print ()

6.36.1.5 random()

Procedure, public mqc_algebra::mqc_scalar::random ()

6.36.1.6 rval()

Procedure, public mqc_algebra::mqc_scalar::rval ()

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

- [src/mqc_algebra.F03](#)

6.37 mqc_est::mqc_scf_eigenvalues Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_eigenvalues](#)
- Procedure, public [getlabel](#) => [mqc_eigenvalues_array_name](#)
- Procedure, public [addlabel](#) => [mqc_eigenvalues_add_name](#)
- Procedure, public [getblock](#) => [mqc_eigenvalues_output_block](#)
- Procedure, public [power](#) => [mqc_scf_eigenvalues_power](#)
- Procedure, public [at](#) => [mqc_eigenvalues_at](#)

6.37.1 Member Function/Subroutine Documentation

6.37.1.1 addlabel()

Procedure, public mqc_est::mqc_scf_eigenvalues::addlabel ()

6.37.1.2 at()

Procedure, public mqc_est::mqc_scf_eigenvalues::at ()

6.37.1.3 getblock()

Procedure, public mqc_est::mqc_scf_eigenvalues::getblock ()

6.37.1.4 getlabel()

Procedure, public mqc_est::mqc_scf_eigenvalues::getlabel ()

6.37.1.5 power()

Procedure, public mqc_est::mqc_scf_eigenvalues::power ()

6.37.1.6 print()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::print ( )
```

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

- [src/mqc_est.F03](#)

6.38 mqc_est::mqc_scf_integral Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_integral](#)
- Procedure, public [getlabel](#) => [mqc_integral_array_name](#)
- Procedure, public [addlabel](#) => [mqc_integral_add_name](#)
- Procedure, public [getblock](#) => [mqc_integral_output_block](#)
- Procedure, public [identity](#) => [mqc_integral_identity](#)
- Procedure, public [init](#) => [mqc_integral_initialize](#)
- Procedure, public [diag](#) => [mqc_scf_integral_diagonalize](#)
- Procedure, public [eigensys](#) => [mqc_scf_integral_generalized_eigensystem](#)
- Procedure, public [inv](#) => [mqc_scf_integral_inverse](#)
- Procedure, public [trace](#) => [mqc_scf_integral_trace](#)
- Procedure, public [det](#) => [mqc_scf_integral_determinant](#)
- Procedure, public [norm](#) => [mqc_integral_norm](#)
- Procedure, public [setelist](#) => [mqc_integral_set_energy_list](#)
- Procedure, public [getelist](#) => [mqc_integral_get_energy_list](#)
- Procedure, public [deleteelist](#) => [mqc_integral_delete_energy_list](#)
- Procedure, public [orbitals](#) => [mqc_integral_output_orbitals](#)
- Procedure, public [swap](#) => [mqc_integral_swap_orbitals](#)

6.38.1 Member Function/Subroutine Documentation

6.38.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_integral::addlabel ( )
```

6.38.1.2 deleteelist()

```
Procedure, public mqc_est::mqc_scf_integral::deleteelist ( )
```

6.38.1.3 det()

Procedure, public mqc_est::mqc_scf_integral::det ()

6.38.1.4 diag()

Procedure, public mqc_est::mqc_scf_integral::diag ()

6.38.1.5 eigensys()

Procedure, public mqc_est::mqc_scf_integral::eigensys ()

6.38.1.6 getblock()

Procedure, public mqc_est::mqc_scf_integral::getblock ()

6.38.1.7 getelist()

Procedure, public mqc_est::mqc_scf_integral::getelist ()

6.38.1.8 getlabel()

Procedure, public mqc_est::mqc_scf_integral::getlabel ()

6.38.1.9 identity()

Procedure, public mqc_est::mqc_scf_integral::identity ()

6.38.1.10 init()

Procedure, public mqc_est::mqc_scf_integral::init ()

6.38.1.11 inv()

Procedure, public mqc_est::mqc_scf_integral::inv ()

6.38.1.12 norm()

Procedure, public mqc_est::mqc_scf_integral::norm ()

6.38.1.13 orbitals()

Procedure, public mqc_est::mqc_scf_integral::orbitals ()

6.38.1.14 print()

Procedure, public mqc_est::mqc_scf_integral::print ()

6.38.1.15 setelist()

Procedure, public mqc_est::mqc_scf_integral::setelist ()

6.38.1.16 swap()

Procedure, public mqc_est::mqc_scf_integral::swap ()

6.38.1.17 trace()

```
Procedure, public mqc_est::mqc_scf_integral::trace ( )
```

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

- [src/mqc_est.F03](#)

6.39 mqc_algebra::mqc_set_array2vector Interface Reference**Public Member Functions**

- subroutine [mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)

6.39.1 Member Function/Subroutine Documentation**6.39.1.1 mqc_set_array2vector_complex()**

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_complex (
    type(mqc_vector), intent(inout) VectorOut,
    complex(kind=real64), dimension(:), intent(in) ArrayIn )
```

6.39.1.2 mqc_set_array2vector_integer()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_integer (
    type(mqc_vector), intent(inout) VectorOut,
    integer(kind=int64), dimension(:), intent(in) ArrayIn )
```

6.39.1.3 mqc_set_array2vector_real()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_real (
    type(mqc_vector), intent(inout) VectorOut,
    real(kind=real64), dimension(:), intent(in) ArrayIn )
```

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

- [src/mqc_algebra.F03](#)

6.40 mqc_est::mqc_twoeris Type Reference

Public Member Functions

- procedure, public [print](#) => [mqc_print_twoeris](#)

6.40.1 Member Function/Subroutine Documentation

6.40.1.1 print()

```
procedure, public mqc_est::mqc_twoeris::print ( )
```

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

- src/[mqc_est.F03](#)

6.41 mqc_algebra::mqc_vector Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_vector_algebra1](#)
- Procedure, public [initialize](#) => [mqc_vector_initialize](#)
- Procedure, public [size](#) => [mqc_length_vector](#)
- Procedure, public [init](#) => [mqc_vector_initialize](#)
- Procedure, public [norm](#) => [mqc_vector_norm](#)
- Procedure, public [transpose](#) => [mqc_vector_transpose](#)
- Procedure, public [dagger](#) => [mqc_vector_conjugate_transpose](#)
- Procedure, public [at](#) => [mqc_vector_scalar_at](#)
- Procedure, public [vat](#) => [mqc_vector_vector_at](#)
- Procedure, public [put](#) => [mqc_vector_scalar_put](#)
- Procedure, public [vput](#) => [mqc_vector_vector_put](#)
- Procedure, public [push](#) => [mqc_vector_push](#)
- Procedure, public [unshift](#) => [mqc_vector_unshift](#)
- Procedure, public [pop](#) => [mqc_vector_pop](#)
- Procedure, public [shift](#) => [mqc_vector_shift](#)
- Procedure, public [maxval](#) => [mqc_vector_maxval](#)
- Procedure, public [minval](#) => [mqc_vector_minloc](#)
- Procedure, public [maxloc](#) => [mqc_vector_maxval](#)
- Procedure, public [minloc](#) => [mqc_vector_minloc](#)
- Procedure, public [argsort](#) => [mqc_vector_argsort](#)
- Procedure, public [sort](#) => [mqc_vector_sort](#)
- Procedure, public [sqrt](#) => [mqc_vector_sqrt](#)
- Procedure, public [abs](#) => [mqc_vector_abs](#)
- Procedure, public [power](#) => [mqc_vector_power](#)
- Procedure, public [diag](#) => [mqc_matrix_diagmatrix_put_vector](#)

Public Attributes

- integer(kind=int64) [length](#) =0
- character(len=64) [data_type](#)
- [real](#)(kind=real64), dimension(:), allocatable [vecr](#)
- integer(kind=int64), dimension(:), allocatable [veci](#)
- complex(kind=real64), dimension(:), allocatable [vecc](#)

6.41.1 Member Function/Subroutine Documentation

6.41.1.1 `abs()`

Procedure, public mqc_algebra::mqc_vector::abs ()

6.41.1.2 `argsort()`

Procedure, public mqc_algebra::mqc_vector::argsort ()

6.41.1.3 `at()`

Procedure, public mqc_algebra::mqc_vector::at ()

6.41.1.4 `dagger()`

Procedure, public mqc_algebra::mqc_vector::dagger ()

6.41.1.5 `diag()`

Procedure, public mqc_algebra::mqc_vector::diag ()

6.41.1.6 init()

Procedure, public mqc_algebra::mqc_vector::init ()

6.41.1.7 initialize()

Procedure, public mqc_algebra::mqc_vector::initialize ()

6.41.1.8 maxloc()

Procedure, public mqc_algebra::mqc_vector::maxloc ()

6.41.1.9 maxval()

Procedure, public mqc_algebra::mqc_vector::maxval ()

6.41.1.10 minloc()

Procedure, public mqc_algebra::mqc_vector::minloc ()

6.41.1.11 minval()

Procedure, public mqc_algebra::mqc_vector::minval ()

6.41.1.12 norm()

Procedure, public mqc_algebra::mqc_vector::norm ()

6.41.1.13 pop()

Procedure, public mqc_algebra::mqc_vector::pop ()

6.41.1.14 power()

Procedure, public mqc_algebra::mqc_vector::power ()

6.41.1.15 print()

Procedure, public mqc_algebra::mqc_vector::print ()

6.41.1.16 push()

Procedure, public mqc_algebra::mqc_vector::push ()

6.41.1.17 put()

Procedure, public mqc_algebra::mqc_vector::put ()

6.41.1.18 shift()

Procedure, public mqc_algebra::mqc_vector::shift ()

6.41.1.19 size()

Procedure, public mqc_algebra::mqc_vector::size ()

6.41.1.20 sort()

```
Procedure, public mqc_algebra::mqc_vector::sort ( )
```

6.41.1.21 sqrt()

```
Procedure, public mqc_algebra::mqc_vector::sqrt ( )
```

6.41.1.22 transpose()

```
Procedure, public mqc_algebra::mqc_vector::transpose ( )
```

6.41.1.23 unshift()

```
Procedure, public mqc_algebra::mqc_vector::unshift ( )
```

6.41.1.24 vat()

```
Procedure, public mqc_algebra::mqc_vector::vat ( )
```

6.41.1.25 vput()

```
Procedure, public mqc_algebra::mqc_vector::vput ( )
```

6.41.2 Member Data Documentation**6.41.2.1 data_type**

```
character(len=64) mqc_algebra::mqc_vector::data_type
```

6.41.2.2 length

```
integer(kind=int64) mqc_algebra::mqc_vector::length =0
```

6.41.2.3 vecc

```
complex(kind=real64), dimension(:), allocatable mqc_algebra::mqc_vector::vecc
```

6.41.2.4 veci

```
integer(kind=int64), dimension(:), allocatable mqc_algebra::mqc_vector::veci
```

6.41.2.5 vecr

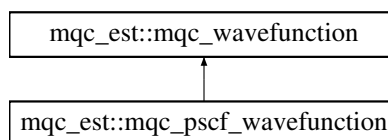
```
real(kind=real64), dimension(:), allocatable mqc_algebra::mqc_vector::vecr
```

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

- [src/mqc_algebra.F03](#)

6.42 mqc_est::mqc_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_wavefunction:



Public Member Functions

- Procedure, public [print](#) => [mqc_print_wavefunction](#)

Public Attributes

- type(mqc_scf_integral) [mo_coefficients](#)
- type(mqc_scf_eigenvalues) [mo_energies](#)
- type(mqc_scf_eigenvalues) [mo_symmetries](#)
- type(mqc_scf_integral) [core_hamiltonian](#)
- type(mqc_scf_integral) [fock_matrix](#)
- type(mqc_scf_integral) [density_matrix](#)
- type(mqc_scf_integral) [scf_density_matrix](#)
- type(mqc_scf_integral) [overlap_matrix](#)
- type(mqc_scalar) [nalpha](#)
- type(mqc_scalar) [nbeta](#)
- type(mqc_scalar) [nelectrons](#)
- type(mqc_scalar) [nbasis](#)
- type(mqc_scalar) [charge](#)
- type(mqc_scalar) [multiplicity](#)
- character(len=256) [basis](#)
- character(len=256) [symmetry](#)
- character(len=256) [wf_type](#)
- logical [wf_complex](#)

6.42.1 Member Function/Subroutine Documentation

6.42.1.1 `print()`

Procedure, public mqc_est::mqc_wavefunction::print ()

6.42.2 Member Data Documentation

6.42.2.1 `basis`

character(len=256) mqc_est::mqc_wavefunction::basis

6.42.2.2 `charge`

type(mqc_scalar) mqc_est::mqc_wavefunction::charge

6.42.2.3 core_hamiltonian

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::core_hamiltonian
```

6.42.2.4 density_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::density_matrix
```

6.42.2.5 fock_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::fock_matrix
```

6.42.2.6 mo_coefficients

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::mo_coefficients
```

6.42.2.7 mo_energies

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_energies
```

6.42.2.8 mo_symmetries

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_symmetries
```

6.42.2.9 multiplicity

```
type(mqc_scalar) mqc_est::mqc_wavefunction::multiplicity
```

6.42.2.10 nalpha

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nalpha
```

6.42.2.11 nbasis

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbasis
```

6.42.2.12 nbeta

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbeta
```

6.42.2.13 nelectrons

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nelectrons
```

6.42.2.14 overlap_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::overlap_matrix
```

6.42.2.15 scf_density_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::scf_density_matrix
```

6.42.2.16 symmetry

```
character(len=256) mqc_est::mqc_wavefunction::symmetry
```

6.42.2.17 wf_complex

```
logical mqc_est::mqc_wavefunction::wf_complex
```

6.42.2.18 wf_type

```
character(len=256) mqc_est::mqc_wavefunction::wf_type
```

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

- [src/mqc_est.F03](#)

6.43 mqc_algebra::operator(*) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalarmultiply](#) (Scalar1, Scalar2)
MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_realscalarmultiply](#) (RealIn, Scalar)
MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarrealmultiply](#) (Scalar, RealIn)
MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type([mqc_vector](#)) function [mqc_scalarvectorproduct](#) (Scalar, Vector)
- type([mqc_vector](#)) function [mqc_vectorscalarproduct](#) (vector, scalar)
- type([mqc_matrix](#)) function [mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type([mqc_matrix](#)) function [mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type([mqc_vector](#)) function [mqc_realvectorproduct](#) (RealIn, Vector)
- type([mqc_vector](#)) function [mqc_vectorrealproduct](#) (vector, realIn)
- type([mqc_vector](#)) function [mqc_integervectorproduct](#) (intIn, Vector)
- type([mqc_vector](#)) function [mqc_vectorintegerproduct](#) (vector, intIn)
- type([mqc_vector](#)) function [mqc_complexvectorproduct](#) (Compln, Vector)
- type([mqc_vector](#)) function [mqc_vectorcomplexproduct](#) (vector, compln)
- type([mqc_matrix](#)) function [mqc_matrixmatrixproduct](#) (MA, MB)

6.43.1 Member Function/Subroutine Documentation

6.43.1.1 `mqc_complexscalarmultiply()`

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_complexscalarmultiply (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply

Author

L. M. Thompson

Date

2019

6.43.1.2 `mqc_complexvectorproduct()`

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_complexvectorproduct (
    complex(kind=real64), intent(in) CompIn,
    type(mqc_vector), intent(in) Vector )
```


6.43.1.3 mqc_integerscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_integerscalarmultiply (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

6.43.1.4 mqc_integervectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_integervectorproduct (
    integer(kind=int64), intent(in) intIn,
    type(mqc_vector), intent(in) Vector )
```

6.43.1.5 mqc_matrixmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::operator(*)::mqc_matrixmatrixproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

6.43.1.6 mqc_matrixscalarproduct()

```
type(mqc_matrix) function mqc_algebra::operator(*)::mqc_matrixscalarproduct (
    type(mqc_matrix), intent(in) Matrix,
    type(mqc_scalar), intent(in) Scalar )
```

6.43.1.7 mqc_realscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_realscalarmultiply (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

6.43.1.8 mqc_realvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_realvectorproduct (
    real(kind=real64), intent(in) RealIn,
    type(mqc_vector), intent(in) Vector )
```

6.43.1.9 mqc_scalarcomplexmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarcomplexmultiply (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply

Author

L. M. Thompson

Date

2019

6.43.1.10 mqc_scalarintegermultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarintegermultiply (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar varibale to multiply

Author

L. M. Thompson

Date

2019

6.43.1.11 mqc_scalarmatrixproduct()

```

type(mqc_matrix) function mqc_algebra::operator(*)::mqc_scalarmatrixproduct (
    type(mqc_scalar), intent(in) Scalar,
    type(mqc_matrix), intent(in) Matrix )

```

6.43.1.12 mqc_scalarmultiply()

```

type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarmultiply (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )

```

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects**Purpose:**

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be multiplied
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be multiplied

Author

L. M. Thompson

Date

2016

6.43.1.13 mqc_scalarrealmultiply()

```

type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarrealmultiply (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )

```

MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to multiply
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply

Author

L. M. Thompson

Date

2019

6.43.1.14 mqc_scalarvectorproduct()

```

type(mqc_vector) function mqc_algebra::operator(*)::mqc_scalarvectorproduct (
    type(mqc_scalar), intent(in) Scalar,
    type(mqc_vector), intent(in) Vector )

```

6.43.1.15 mqc_vectorcomplexproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorcomplexproduct (
    type(mqc_vector), intent(in) vector,
    complex(kind=real64), intent(in) compIn )
```

6.43.1.16 mqc_vectorintegerproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorintegerproduct (
    type(mqc_vector), intent(in) vector,
    integer(kind=int64), intent(in) intIn )
```

6.43.1.17 mqc_vectorrealproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorrealproduct (
    type(mqc_vector), intent(in) vector,
    real(kind=real64), intent(in) realIn )
```

6.43.1.18 mqc_vectorscalarproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorscalarproduct (
    type(mqc_vector), intent(in) vector,
    type(mqc_scalar), intent(in) scalar )
```

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

- src/[mqc_algebra.F03](#)

6.44 mqc_est::operator(*) Interface Reference

Public Member Functions

- type([mqc_scf_integral](#)) function [mqc_scalar_integral_multiply](#) (scalar, integral)
- type([mqc_scf_integral](#)) function [mqc_integral_scalar_multiply](#) (integral, scalar)

6.44.1 Member Function/Subroutine Documentation

6.44.1.1 mqc_integral_scalar_multiply()

```
type(mqc_scf_integral) function mqc_est::operator(*)::mqc_integral_scalar_multiply (
    type(mqc_scf_integral), intent(in) integral,
    type(mqc_scalar), intent(in) scalar )
```

6.44.1.2 mqc_scalar_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::operator(*)::mqc_scalar_integral_multiply (
    type(mqc_scalar), intent(in) scalar,
    type(mqc_scf_integral), intent(in) integral )
```

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

- [src/mqc_est.F03](#)

6.45 mqc_algebra::operator(**) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalarexponent](#) (Scalar1, Scalar2)
MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarintegerexponent](#) (Scalar, IntIn)
MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an intrinsic integer
- type([mqc_scalar](#)) function [mqc_scalarrealexponent](#) (Scalar, RealIn)
MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real
- type([mqc_scalar](#)) function [mqc_scalarcomplexexponent](#) (Scalar, Compln)
MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex

6.45.1 Member Function/Subroutine Documentation

6.45.1.1 mqc_scalarcomplexexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarcomplexexponent (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) CompIn )
```

MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex

Purpose:

MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex.

Parameters

in	<i>Scalar</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>Comp↔ In</i>	CompIn is Complex(kind=real64) The power value

Author

L. M. Thompson

Date

2019

6.45.1.2 mqc_scalarexponent()

```

type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarexponent (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )

```

MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar

Purpose:

MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The power value

Author

L. M. Thompson

Date

2016

6.45.1.3 mqc_scalarintegerexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarintegerexponent (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarIntegerExponent is a function that raises an **MQC_Scalar** to the power of an intrinsic integer

Purpose:

MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The power value

Author

L. M. Thompson

Date

2019

6.45.1.4 mqc_scalarrealexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarrealexponent (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealExponent is a function that raises an **MQC_Scalar** to the power of an intrinsic real

Purpose:

MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar1 is Type(MQC_Scalar) The base value
in	<i>RealIn</i>	RealIn is Real(kind=real64) The power value

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.46 mqc_est::operator(+) Interface Reference

Public Member Functions

- `type(mqc_scf_integral)` function `mqc_integral_sum` (integralA, integralB)

6.46.1 Member Function/Subroutine Documentation

6.46.1.1 mqc_integral_sum()

```
type(mqc_scf_integral) function mqc_est::operator(+):mqc_integral_sum (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), intent(in) integralB )
```

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

- [src/mqc_est.F03](#)

6.47 mqc_algebra::operator(+) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalaradd](#) (Scalar1, Scalar2)
MQC_ScalarAdd is a function that sums two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalaradd](#) (IntegerIn, Scalar)
MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarintegeradd](#) (Scalar, IntegerIn)
MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_realscalaradd](#) (RealIn, Scalar)
MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarrealadd](#) (Scalar, RealIn)
MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_complexscalaradd](#) (ComplexIn, Scalar)
MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarcomplexadd](#) (Scalar, ComplexIn)
MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type([mqc_vector](#)) function [mqc_vectorvectorsum](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_scalarvectorsum](#) (ScalarIn, VectorIn)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsum](#) (MA, MB)

6.47.1 Member Function/Subroutine Documentation

6.47.1.1 mqc_complexscalaradd()

```
type(mqc\_scalar) function mqc_algebra::operator(+)::mqc_complexscalaradd (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc\_scalar), intent(in) Scalar )
```

MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	Complex is Complex(kind=real64) The intrinsic complex variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum
Generated by Doxygen		

Author

L. M. Thompson

Date

2019

6.47.1.2 mqc_integerscalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_integerscalaradd (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

MQC_IntegerScalarAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

6.47.1.3 mqc_matrixmatrixsum()

```
type(mqc_matrix) function mqc_algebra::operator(+)::mqc_matrixmatrixsum (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

6.47.1.4 mqc_realscalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_realscalaradd (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

6.47.1.5 mqc_scalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalaradd (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

Purpose:

MQC_ScalarAdd is a function that sums two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be summed
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be summed

Author

L. M. Thompson

Date

2016

6.47.1.6 mqc_scalarcomplexadd()

```

type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarcomplexadd (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )

```

MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>Complex↔ In</i>	Complex is Complex(kind=real64) The intrinsic complex variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum

Author

L. M. Thompson

Date

2019

6.47.1.7 mqc_scalarintegeradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarintegeradd (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar

Purpose:

MQC_ScalarIntegerSum is a function that is used to sum an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar varibale to sum

Author

L. M. Thompson

Date

2019

6.47.1.8 mqc_scalarrealadd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarrealadd (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

MQC_ScalarRealSum is a function that is used to sum an intrinsic real by an MQC_Scalar.

Parameters

in	<i>realIn</i>	RealIn is Real(kind=real64) The intrinsic real variable to sum
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum

Author

L. M. Thompson

Date

2019

6.47.1.9 mqc_scalarvectorsum()

```

type(mqc_vector) function mqc_algebra::operator(+)::mqc_scalarvectorsum (
    type(mqc_scalar), intent(in) ScalarIn,
    type(mqc_vector), intent(in) VectorIn )

```

6.47.1.10 mqc_vectorvectorsum()

```

type(mqc_vector) function mqc_algebra::operator(+)::mqc_vectorvectorsum (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )

```

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

- src/[mqc_algebra.F03](#)

6.48 mqc_est::operator(-) Interface Reference**Public Member Functions**

- type([mqc_scf_integral](#)) function [mqc_integral_difference](#) (integralA, integralB)

6.48.1 Member Function/Subroutine Documentation

6.48.1.1 mqc_integral_difference()

```
type(mqc_scf_integral) function mqc_est::operator(-)::mqc_integral_difference (
    type(mqc_scf_integral), intent(in) integralA,
    type(mqc_scf_integral), intent(in) integralB )
```

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

- [src/mqc_est.F03](#)

6.49 mqc_algebra::operator(-) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalarsubtract](#) (Scalar1, Scalar2)
MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalarsubtract](#) (IntegerIn, Scalar)
MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic integer
- type([mqc_scalar](#)) function [mqc_scalarintegersubtract](#) (Scalar, IntegerIn)
MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_realscalarsubtract](#) (RealIn, Scalar)
MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic real
- type([mqc_scalar](#)) function [mqc_scalarrealsubtract](#) (Scalar, RealIn)
MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar
- type([mqc_scalar](#)) function [mqc_complexscalarsubtract](#) (ComplexIn, Scalar)
MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic complex
- type([mqc_scalar](#)) function [mqc_scalarcomplexsubtract](#) (Scalar, ComplexIn)
MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar
- type([mqc_vector](#)) function [mqc_vectorvectordifference](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_scalarvectordifference](#) (ScalarIn, VectorIn)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsubtract](#) (MA, MB)

6.49.1 Member Function/Subroutine Documentation

6.49.1.1 mqc_complexscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_complexscalarsubtract (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic complex

Purpose:

MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic complex.

Parameters

in	<i>Complex</i> ↔ <i>In</i>	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

6.49.1.2 mqc_integerscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_integerscalarsubtract (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic integer

Purpose:

MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic integer.

Parameters

in	<i>Integer</i> ↔ <i>In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

6.49.1.3 mqc_matrixmatrixsubtract()

```

type(mqc_matrix) function mqc_algebra::operator(-)::mqc_matrixmatrixsubtract (
    type(mqc_matrix), intent(in)  MA,
    type(mqc_matrix), intent(in)  MB )

```

6.49.1.4 mqc_realscalarsubtract()

```

type(mqc_scalar) function mqc_algebra::operator(-)::mqc_realscalarsubtract (
    real(kind=real64), intent(in)  RealIn,
    type(mqc_scalar), intent(in)  Scalar )

```

MQC_RealScalarSubtract is a function that is used to subtract an **MQC_Scalar** from an intrinsic real

Purpose:

MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinsic real.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real to subtract from
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract

Author

L. M. Thompson

Date

2019

6.49.1.5 mqc_scalarcomplexsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarcomplexsubtract (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar

Purpose:

MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>Complex↔ In</i>	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract

Author

L. M. Thompson

Date

2019

6.49.1.6 mqc_scalarintegersubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarintegersubtract (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar

Purpose:

MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract

Author

L. M. Thompson

Date

2019

6.49.1.7 mqc_scalarrealsubtract()

```

type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarrealsubtract (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )

```

MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar

Purpose:

MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from
in	<i>Real↔ In</i>	RealIn is Real(kind=real64) The intrinsic real to subtract

Author

L. M. Thompson

Date

2019

6.49.1.8 mqc_scalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarsubtract (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects**Purpose:**

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar from which Scalar2 will be subtracted
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar which will be subtracted from Scalar1

Author

L. M. Thompson

Date

2016

6.49.1.9 mqc_scalarvectordifference()

```
type(mqc_vector) function mqc_algebra::operator(-)::mqc_scalarvectordifference (
    type(mqc_scalar), intent(in) ScalarIn,
    type(mqc_vector), intent(in) VectorIn )
```

6.49.1.10 mqc_vectorvectordifference()

```
type(mqc_vector) function mqc_algebra::operator(-)::mqc_vectorvectordifference (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

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

- [src/mqc_algebra.F03](#)

6.50 mqc_algebra::operator(.dot.) Interface Reference

Public Member Functions

- type(mqc_scalar) function [mqc_vectorvectordotproduct](#) (Vector1, Vector2)
- type(mqc_vector) function [mqc_vectormatrixdotproduct](#) (VA, MB)
- type(mqc_vector) function [mqc_matrixvectordotproduct](#) (MA, VB)
- type(mqc_matrix) function [mqc_matrixmatrixdotproduct](#) (MA, MB)

6.50.1 Member Function/Subroutine Documentation

6.50.1.1 mqc_matrixmatrixdotproduct()

```
type(mqc_matrix) function mqc_algebra::operator(.dot.)::mqc_matrixmatrixdotproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_matrix), intent(in) MB )
```

6.50.1.2 mqc_matrixvectordotproduct()

```
type(mqc_vector) function mqc_algebra::operator(.dot.)::mqc_matrixvectordotproduct (
    type(mqc_matrix), intent(in) MA,
    type(mqc_vector), intent(in) VB )
```

6.50.1.3 mqc_vectormatrixdotproduct()

```
type(mqc_vector) function mqc_algebra::operator(.dot.)::mqc_vectormatrixdotproduct (
    type(mqc_vector), intent(in) VA,
    type(mqc_matrix), intent(in) MB )
```

6.50.1.4 `mqc_vectorvectordotproduct()`

```
type(mqc_scalar) function mqc_algebra::operator(.dot.)::mqc_vectorvectordotproduct (
    type(mqc_vector), intent(in) Vector1,
    type(mqc_vector), intent(in) Vector2 )
```

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

- [src/mqc_algebra.F03](#)

6.51 `mqc_algebra::operator(.eq.)` Interface Reference

Public Member Functions

- logical function [mqc_scalareq](#) (Scalar1, Scalar2)
MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal

6.51.1 Member Function/Subroutine Documentation

6.51.1.1 `mqc_scalareq()`

```
logical function mqc_algebra::operator(.eq.)::mqc_scalareq (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal

Purpose:

MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

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

- [src/mqc_algebra.F03](#)

6.52 `mqc_algebra::operator(.ewd.)` Interface Reference

Public Member Functions

- `type(mqc_matrix)` function [mqc_elementmatrixdivide](#) (A, B)

6.52.1 Member Function/Subroutine Documentation

6.52.1.1 `mqc_elementmatrixdivide()`

```
type(mqc_matrix) function mqc_algebra::operator(.ewd.)::mqc_elementmatrixdivide (
    type(mqc_matrix), intent(in) A,
    type(mqc_matrix), intent(in) B )
```

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

- [src/mqc_algebra.F03](#)

6.53 `mqc_algebra::operator(.ewp.)` Interface Reference

Public Member Functions

- `type(mqc_vector)` function [mqc_elementvectorproduct](#) (Vector1In, Vector2In)
- `type(mqc_matrix)` function [mqc_elementmatrixproduct](#) (A, B)

6.53.1 Member Function/Subroutine Documentation

6.53.1.1 mqc_elementmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::operator(.ewp.)::mqc_elementmatrixproduct (
    type(mqc_matrix), intent(in) A,
    type(mqc_matrix), intent(in) B )
```

6.53.1.2 mqc_elementvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(.ewp.)::mqc_elementvectorproduct (
    type(mqc_vector), intent(in) Vector1In,
    type(mqc_vector), intent(in) Vector2In )
```

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

- src/[mqc_algebra.F03](#)

6.54 mqc_algebra::operator(.ge.) Interface Reference

Public Member Functions

- logical function [mqc_scalarge](#) (Scalar1, Scalar2)

***MQC_ScalarGE** is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_Scalar*

6.54.1 Member Function/Subroutine Documentation

6.54.1.1 mqc_scalarge()

```
logical function mqc_algebra::operator(.ge.)::mqc_scalarge (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_Scalar

Purpose:

MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal to the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is is greater than or equal to the right real part and FALSE if the left real part is less than the right real part.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

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

- [src/mqc_algebra.F03](#)

6.55 `mqc_algebra::operator(.gt.)` Interface Reference

Public Member Functions

- logical function [mqc_scalargt](#) (Scalar1, Scalar2)
MQC_ScalarGT is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar
- logical function [mqc_scalargtinteger](#) (Scalar, IntIn)
MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer
- logical function [mqc_integertscalar](#) (IntIn, Scalar)
MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar
- logical function [mqc_scalargtreal](#) (Scalar, RealIn)
MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real
- logical function [mqc_realgtscalar](#) (RealIn, Scalar)
MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar

6.55.1 Member Function/Subroutine Documentation

6.55.1.1 `mqc_integertgtscalar()`

```
logical function mqc_algebra::operator(.gt.)::mqc_integertgtscalar (
    integer(kind=int64), intent(in) IntIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar

Purpose:

MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer is greater than the real part of the MQC_Scalar and FALSE if the intrinsic integer is less than the real part of the MQC_Scalar. If the intrinsic integer is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise.

Parameters

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.55.1.2 `mqc_realgtgtscalar()`

```
logical function mqc_algebra::operator(.gt.)::mqc_realgtgtscalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar

Purpose:

`MQC_RealGTScalar` is a function that returns `TRUE` if an intrinsic real is greater than a `MQC_Scalar`.

When dealing with complex numbers, the function returns `TRUE` if the intrinsic real is greater than the real part of the `MQC_Scalar` and `FALSE` if the intrinsic real is less than the real part of the `MQC_Scalar`. If the intrinsic real is equal to the real part of the `MQC_Scalar`, the function returns `TRUE` if the imaginary part of `MQC_Scalar` is less than zero and `FALSE` otherwise.

Parameters

in	<i>RealIn</i>	RealIn is <code>Real(kind=real64)</code> The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> that will be tested.

Author

L. M. Thompson

Date

2019

6.55.1.3 mqc_scalargt()

```
logical function mqc_algebra::operator(.gt.)::mqc_scalargt (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGT is a function that returns `TRUE` if the left `MQC_Scalar` is greater than the right `MQC_Scalar`

Purpose:

`MQC_ScalarGT` is a function that returns `TRUE` if the left `MQC_Scalar` is greater than the right `MQC_Scalar`.

When dealing with complex numbers, the function returns `TRUE` if the left real part is greater than the right real part and `FALSE` if the left real part is less than the right real part. If the left real part is equal to the right real part, the function returns `TRUE` if the left imaginary part is greater than the right imaginary part and `FALSE` otherwise.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.55.1.4 mqc_scalargtinteger()

```
logical function mqc_algebra::operator(.gt.):mqc_scalargtinteger (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer

Purpose:

MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is greater than the intrinsic integer and FALSE if the real part of the MQC_Scalar is less than the intrinsic integer. If the real part of the MQC_Scalar is equal to the intrinsic integer, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.

Author

L. M. Thompson

Date

2019

6.55.1.5 `mqc_scalargtreal()`

```
logical function mqc_algebra::operator(.gt.)::mqc_scalargtreal (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real

Purpose:

MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is greater than the intrinsic real and FALSE if the real part of the MQC_Scalar is less than the intrinsic real. If the real part of the MQC_Scalar is equal to the intrinsic real, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	RealIn is Real(kind=int64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.56 mqc_algebra::operator(.le.) Interface Reference

Public Member Functions

- logical function [mqc_scalarle](#) (Scalar1, Scalar2)
MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar
- logical function [mqc_scalarlereal](#) (Scalar, Realln)
MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real
- logical function [mqc_reallescalar](#) (Realln, Scalar)
MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar
- logical function [mqc_scalarleinteger](#) (Scalar, Intln)
MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer
- logical function [mqc_integerlescalar](#) (Intln, Scalar)
MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar

6.56.1 Member Function/Subroutine Documentation

6.56.1.1 mqc_integerlescalar()

```
logical function mqc_algebra::operator(.le.)::mqc_integerlescalar (
    integer(kind=int64), intent(in) IntIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar ↵

Purpose:

MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic integer is greater than the real part of the MQC_Scalar.

Parameters

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.56.1.2 `mqc_reallescalar()`

```
logical function mqc_algebra::operator(.le.)::mqc_reallescalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar

Purpose:

MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar.

Parameters

in	<i>Real↔ In</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.56.1.3 mqc_scalarle()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarle (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar

Purpose:

MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal to the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is less than or equal to the right real part and FALSE if the left real part is greater than the right real part.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.56.1.4 mqc_scalarleinteger()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarleinteger (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer

Purpose:

MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than or equal to the intrinsic integer and FALSE if the real part of the MQC_Scalar is greater than the intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.

Author

L. M. Thompson

Date

2019

6.56.1.5 mqc_scalarlereal()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarlereal (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLereal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real

Purpose:

MQC_ScalarLereal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than or equal to the intrinsic real and FALSE if the real part of the MQC_Scalar is greater than the intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>IntIn</i>	RealIn is Real(kind=int64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.57 mqc_algebra::operator(.lt.) Interface Reference

Public Member Functions

- logical function [mqc_scalarlt](#) (Scalar1, Scalar2)
MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar
- logical function [mqc_scalarltreal](#) (Scalar, RealIn)
MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real
- logical function [mqc_realltscalar](#) (RealIn, Scalar)
MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

6.57.1 Member Function/Subroutine Documentation

6.57.1.1 mqc_realltscalar()

```
logical function mqc_algebra::operator(.lt.)::mqc_realltscalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

Purpose:

MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar. If the intrinsic real is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise.

Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.57.1.2 mqc_scalarlt()

```
logical function mqc_algebra::operator(.lt.)::mqc_scalarlt (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar

Purpose:

MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is less than the right real part and FALSE if the left real part is greater than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is less than the right imaginary part and FALSE otherwise.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.57.1.3 mqc_scalarltreal()

```
logical function mqc_algebra::operator(.lt.)::mqc_scalarltreal (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real

Purpose:

MQC_ScalarLReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than the intrinsic real and FALSE if the real part of the MQC_Scalar is greater than the intrinsic real. If the real part of the MQC_Scalar is equal to the intrinsic real, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.58 mqc_algebra::operator(.ne.) Interface Reference

Public Member Functions

- logical function [mqc_scalarne](#) (Scalar1, Scalar2)

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal

6.58.1 Member Function/Subroutine Documentation

6.58.1.1 mqc_scalarne()

```
logical function mqc_algebra::operator(.ne.)::mqc_scalarne (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal

Purpose:

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

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

- src/[mqc_algebra.F03](#)

6.59 mqc_algebra::operator(.outer.) Interface Reference

Public Member Functions

- type([mqc_matrix](#)) function [mqc_outer](#) (VA, VB)

6.59.1 Member Function/Subroutine Documentation

6.59.1.1 mqc_outer()

```
type(mqc\_matrix) function mqc_algebra::operator(.outer.)::mqc_outer (
    type(mqc\_vector), intent(in) VA,
    type(mqc\_vector), intent(in) VB )
```

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

- src/[mqc_algebra.F03](#)

6.60 mqc_algebra::operator(.x.) Interface Reference

Public Member Functions

- type([mqc_vector](#)) function [mqc_crossproduct](#) (Vector1In, Vector2In)

6.60.1 Member Function/Subroutine Documentation

6.60.1.1 mqc_crossproduct()

```
type(mqc\_vector) function mqc_algebra::operator(.x.)::mqc_crossproduct (
    type(mqc\_vector), intent(in) Vector1In,
    type(mqc\_vector), intent(in) Vector2In )
```

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

- src/[mqc_algebra.F03](#)

6.61 `mqc_algebra::operator(/)` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalardivide` (Scalar1, Scalar2)
MQC_ScalarDivide is a function that divides two MQC_Scalar objects
- `type(mqc_scalar)` function `mqc_integerscalardivide` (IntegerIn, Scalar)
MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar
- `type(mqc_scalar)` function `mqc_scalarintegerdivide` (Scalar, IntegerIn)
MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer
- `type(mqc_scalar)` function `mqc_realscalardivide` (RealIn, Scalar)
MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar
- `type(mqc_scalar)` function `mqc_scalarrealddivide` (Scalar, RealIn)
MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an intrinsic real
- `type(mqc_scalar)` function `mqc_complexscalardivide` (ComplexIn, Scalar)
MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar
- `type(mqc_scalar)` function `mqc_scalarcomplexdivide` (Scalar, ComplexIn)
MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex
- `type(mqc_vector)` function `mqc_vectorscalardivide` (vector, scalar)
- `type(mqc_vector)` function `mqc_vectorrealddivide` (vector, realIn)
- `type(mqc_vector)` function `mqc_vectorintegerdivide` (vector, intIn)
- `type(mqc_vector)` function `mqc_vectorcomplexdivide` (vector, compIn)

6.61.1 Member Function/Subroutine Documentation

6.61.1.1 `mqc_complexscalardivide()`

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_complexscalardivide (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar.

Parameters

in	<i>ComplexIn</i>	ComplexIn is Complex(kind=real64) The intrinsic complex variable numerator
in	<i>Scalar</i>	
Generated by Doxygen		Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.2 mqc_integerscalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_integerscalardivide (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar

Purpose:

MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar.

Parameters

in	<i>IntegerIn</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable numerator
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.3 mqc_realscalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_realscalardivide (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar

Purpose:

MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar.

Parameters

in	<i>Real↔ In</i>	RealIn is Real(kind=real64) The intrinsic real variable numerator
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.4 mqc_scalarcomplexdivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarcomplexdivide (
    type(mqc_scalar), intent(in) Scalar,
    complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex

Purpose:

MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator
in	<i>Complex↔ In</i>	ComplexIn is Complex(kind=real64) The intrinsic complex variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.5 mqc_scalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalardivide (
    type(mqc_scalar), intent(in) Scalar1,
    type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarDivide is a function that divides two **MQC_Scalar** objects

Purpose:

MQC_ScalarDivide is a function that divides MQC_Scalar objects.

Parameters

in	<i>Scalar1</i>	Scalar1 is Type(MQC_Scalar) The numerator
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The denominator

Author

L. M. Thompson

Date

2016

6.61.1.6 mqc_scalarintegerdivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarintegerdivide (
    type(mqc_scalar), intent(in) Scalar,
    integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerDivide is a function that is used to divide an **MQC_Scalar** by an intrinsic integer

Purpose:

`MQC_ScalarIntegerDivide` is a function that is used to divide an `MQC_Scalar` by an intrinsic integer.

Parameters

in	<i>Scalar</i>	Scalar is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> variable numerator
in	<i>IntegerIn</i>	<code>IntegerIn</code> is <code>Integer(kind=int64)</code> The intrinsic integer variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.7 mqc_scalarrealddivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarrealddivide (
    type(mqc_scalar), intent(in) Scalar,
    real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealDivide is a function that is used to divide an `MQC_Scalar` by an intrinsic real

Purpose:

`MQC_ScalarRealDivide` is a function that is used to divide an `MQC_Scalar` by an intrinsic real.

Parameters

in	<i>Scalar</i>	Scalar is <code>Type(MQC_Scalar)</code> The <code>MQC_Scalar</code> variable numerator
in	<i>RealIn</i>	<code>RealIn</code> is <code>Real(kind=real64)</code> The intrinsic real variable denominator

Author

L. M. Thompson

Date

2019

6.61.1.8 mqc_vectorcomplexdivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorcomplexdivide (
    type(mqc_vector), intent(in) vector,
    complex(kind=real64), intent(in) compIn )
```

6.61.1.9 mqc_vectorintegerdivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorintegerdivide (
    type(mqc_vector), intent(in) vector,
    integer(kind=int64), intent(in) intIn )
```

6.61.1.10 mqc_vectorrealddivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorrealddivide (
    type(mqc_vector), intent(in) vector,
    real(kind=real64), intent(in) realIn )
```

6.61.1.11 mqc_vectorscalarddivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorscalarddivide (
    type(mqc_vector), intent(in) vector,
    type(mqc_scalar), intent(in) scalar )
```

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

- [src/mqc_algebra.F03](#)

6.62 mqc_algebra::real Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_complex_realpart](#) (ScalarIn)
MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar
- type([mqc_vector](#)) function [mqc_vector_complex_realpart](#) (A)

6.62.1 Member Function/Subroutine Documentation

6.62.1.1 mqc_scalar_complex_realpart()

```
type(mqc\_scalar) function mqc_algebra::real::mqc_scalar_complex_realpart (
    type(mqc\_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar

Purpose:

MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar.

Parameters

in	<i>ScalarIn</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable
----	-----------------	---

Author

L. M. Thompson

Date

2019

6.62.1.2 mqc_vector_complex_realpart()

```
type(mqc\_vector) function mqc_algebra::real::mqc_vector_complex_realpart (
    class(mqc\_vector), intent(in) A )
```

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

- src/[mqc_algebra.F03](#)

6.63 mqc_algebra::sin Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_sin` (Scalar)
MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

6.63.1 Member Function/Subroutine Documentation

6.63.1.1 mqc_scalar_sin()

```
type(mqc_scalar) function mqc_algebra::sin::mqc_scalar_sin (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

Purpose:

MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar.

Parameters

in	Scalar	Scalar is Type(MQC_Scalar) The argument of the function
----	--------	--

Author

L. M. Thompson

Date

2019

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

- `src/mqc_algebra.F03`

6.64 mqc_algebra::sqrt Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_sqrt` (Scalar)
MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

6.64.1 Member Function/Subroutine Documentation

6.64.1.1 mqc_scalar_sqrt()

```
type(mqc_scalar) function mqc_algebra::sqrt::mqc_scalar_sqrt (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sqrt is a function used to return the square root of an **MQC_scalar**

Purpose:

MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar.

Parameters

in	Scalar	Scalar is Type(MQC_Scalar) The argument of the function
----	--------	--

Author

L. M. Thompson

Date

2016

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

- src/[mqc_algebra.F03](#)

6.65 mqc_algebra::tan Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_tan](#) (Scalar)
MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar

6.65.1 Member Function/Subroutine Documentation

6.65.1.1 `mqc_scalar_tan()`

```
type(mqc_scalar) function mqc_algebra::tan::mqc_scalar_tan (
    type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Tan is a function used to return the tangent of an **MQC_scalar**

Purpose:

`MQC_Scalar_Tan` is a function used to return the tangent of an `MQC_scalar`.

Parameters

<code>in</code>	<i>Scalar</i>	<p><code>Scalar</code> is <code>Type(MQC_Scalar)</code> The argument of the function</p>
-----------------	---------------	--

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.66 `mqc_est::transpose` Interface Reference

Public Member Functions

- type(`mqc_scf_integral`) function `mqc_integral_transpose` (`integral`, `label`)

6.66.1 Member Function/Subroutine Documentation

6.66.1.1 `mqc_integral_transpose()`

```
type(mqc_scf_integral) function mqc_est::transpose::mqc_integral_transpose (
    type(mqc_scf_integral), intent(in) integral,
    character(len=*), intent(in), optional label )
```

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

- [src/mqc_est.F03](#)

6.67 `mqc_algebra::transpose` Interface Reference

Public Member Functions

- type(`mqc_vector`) function [mqc_vector_transpose](#) (`Vector`)
- type(`mqc_matrix`) function [mqc_matrix_transpose](#) (`Matrix`)

6.67.1 Member Function/Subroutine Documentation

6.67.1.1 `mqc_matrix_transpose()`

```
type(mqc_matrix) function mqc_algebra::transpose::mqc_matrix_transpose (
    class(mqc_matrix), intent(in) Matrix )
```

6.67.1.2 `mqc_vector_transpose()`

```
type(mqc_vector) function mqc_algebra::transpose::mqc_vector_transpose (
    class(mqc_vector), intent(in) Vector )
```

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

- [src/mqc_algebra.F03](#)

Chapter 7

File Documentation

7.1 src/mqc_algebra.F03 File Reference

Data Types

- type [mqc_algebra::mqc_scalar](#)
- type [mqc_algebra::mqc_vector](#)
- type [mqc_algebra::mqc_matrix](#)
- type [mqc_algebra::mqc_r4tensor](#)
- interface [mqc_algebra::mqc_print](#)
- interface [mqc_algebra::contraction](#)
- interface [mqc_algebra::conjg](#)
- interface [mqc_algebra::mqc_have_real](#)
- interface [mqc_algebra::mqc_have_int](#)
- interface [mqc_algebra::mqc_have_complex](#)
- interface [mqc_algebra::mqc_cast_real](#)
- interface [mqc_algebra::mqc_cast_complex](#)
- interface [mqc_algebra::matmul](#)
- interface [mqc_algebra::transpose](#)
- interface [mqc_algebra::dagger](#)
- interface [mqc_algebra::cmplx](#)
- interface [mqc_algebra::sqrt](#)
- interface [mqc_algebra::abs](#)
- interface [mqc_algebra::real](#)
- interface [mqc_algebra::aimag](#)
- interface [mqc_algebra::sin](#)
- interface [mqc_algebra::cos](#)
- interface [mqc_algebra::tan](#)
- interface [mqc_algebra::asin](#)
- interface [mqc_algebra::acos](#)
- interface [mqc_algebra::atan](#)
- interface [mqc_algebra::atan2](#)
- interface [mqc_algebra::mqc_set_array2vector](#)
- interface [mqc_algebra::mqc_matrix_symmmatrix_put](#)

- interface [mqc_algebra::mqc_matrix_diagmatrix_put](#)
- interface [mqc_algebra::matrix_symm2sq](#)
- interface [mqc_algebra::dot_product](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(/\)](#)
- interface [mqc_algebra::operator\(**\)](#)
- interface [mqc_algebra::operator\(.ne.\)](#)
- interface [mqc_algebra::operator\(.eq.\)](#)
- interface [mqc_algebra::operator\(.lt.\)](#)
- interface [mqc_algebra::operator\(.gt.\)](#)
- interface [mqc_algebra::operator\(.le.\)](#)
- interface [mqc_algebra::operator\(.ge.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(.dot.\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(/\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(.ewp.\)](#)
- interface [mqc_algebra::operator\(.ewd.\)](#)
- interface [mqc_algebra::operator\(.x.\)](#)
- interface [mqc_algebra::operator\(.outer.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(.dot.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)

Modules

- module [mqc_algebra](#)

Functions/Subroutines

- integer(kind=int64) function [mqc_algebra::factorial](#) (n)
Factorial returns the factorial of an integer
- integer(kind=int64) function [mqc_algebra::bin_coeff](#) (N, K)
Bin_Coeff returns the binomial coefficient of (n,k)
- subroutine [mqc_algebra::mqc_allocate_scalar](#) (Scalar, Data_type)
MQC_Allocate_Scalar is used to allocate a scalar type variable of the MQC_Scalar class
- subroutine [mqc_algebra::mqc_deallocate_scalar](#) (Scalar)
MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class
- logical function [mqc_algebra::mqc_scalar_isallocated](#) (Scalar)
MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar

- subroutine [mqc_algebra::mqc_input_integer_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Integer_Scalar** is a subroutine is used to set an intrinsic integer to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_input_real_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Real_Scalar** is a subroutine is used to set an intrinsic real to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_input_complex_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Complex_Scalar** is a subroutine is used to set an intrinsic complex to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_mqcscalar_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output MQCScalar_Scalar** is a subroutine used to output an MQC_scalar equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_integer_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Integer_Scalar** is a subroutine used to output an intrinsic integer equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_real_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Real_Scalar** is a subroutine used to output an intrinsic real equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_complex_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Complex_Scalar** is a subroutine used to output an intrinsic complex equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_print_scalar_algebra1](#) (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)
***MQC_Print_Scalar_Algebra1** is a subroutine used to print an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_cmplx](#) (Scalar1, Scalar2)
***MQC_Scalar_Cmplx** is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_sqrt](#) (Scalar)
***MQC_Scalar_Sqrt** is a function used to return the square root of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_sin](#) (Scalar)
***MQC_Scalar_Sin** is a function used to return the sine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_cos](#) (Scalar)
***MQC_Scalar_Cos** is a function used to return the cosine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_tan](#) (Scalar)
***MQC_Scalar_Tan** is a function used to return the tangent of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_asin](#) (Scalar)
***MQC_Scalar_ASin** is a function used to return the arcsin of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_acos](#) (Scalar)
***MQC_Scalar_ACos** is a function used to return the arccosine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_atan](#) (Scalar)
***MQC_Scalar_ATan** is a function used to return the arctangent of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_atan2](#) (Scalar)
***MQC_Scalar_ATan2** is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram*
- logical function [mqc_algebra::mqc_scalar_havereal](#) (Scalar)
***MQC_Scalar_HaveReal** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real*
- logical function [mqc_algebra::mqc_scalar_haveinteger](#) (Scalar)
***MQC_Scalar_HaveInteger** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer*
- logical function [mqc_algebra::mqc_scalar_havecomplex](#) (Scalar)
***MQC_Scalar_HaveComplex** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex*
- real(kind=real64) function [mqc_algebra::mqc_scalar_get_intrinsic_real](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Real** is a function that returns the MQC_scalar value as an intrinsic real*
- integer(kind=int64) function [mqc_algebra::mqc_scalar_get_intrinsic_integer](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Integer** is a function that returns the MQC_scalar value as an intrinsic integer*

- complex(kind=real64) function [mqc_algebra::mqc_scalar_get_intrinsic_complex](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Complex** is a function that returns the MQC_scalar value as an intrinsic complex*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_get_abs_value](#) (Scalar)
***MQC_Scalar_Get_ABS_Value** is a function that returns the absolute value of MQC_scalar variable*
- subroutine [mqc_algebra::mqc_scalar_get_random_value](#) (Scalar)
***MQC_Scalar_Get_Random_Value** is a function that returns a random real value from a uniform distribution between zero and one*
- type(mqc_scalar) function [mqc_algebra::mqc_scalaradd](#) (Scalar1, Scalar2)
***MQC_ScalarAdd** is a function that sums two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarsubtract](#) (Scalar1, Scalar2)
***MQC_ScalarSubtract** is a function that subtracts two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarmultiply](#) (Scalar1, Scalar2)
***MQC_ScalarMultiply** is a function that multiplies two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalardivide](#) (Scalar1, Scalar2)
***MQC_ScalarDivide** is a function that divides two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarexponent](#) (Scalar1, Scalar2)
***MQC_ScalarExponent** is a function that raises one MQC_Scalar to the power of another MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegerexponent](#) (Scalar, IntIn)
***MQC_ScalarIntegerExponent** is a function that raises an MQC_Scalar to the power of an intrinsic integer*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealexponent](#) (Scalar, RealIn)
***MQC_ScalarRealExponent** is a function that raises an MQC_Scalar to the power of an intrinsic real*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexexponent](#) (Scalar, Compln)
***MQC_ScalarComplexExponent** is a function that raises an MQC_Scalar to the power of an intrinsic complex*
- logical function [mqc_algebra::mqc_scalarne](#) (Scalar1, Scalar2)
***MQC_ScalarNE** is a function that returns TRUE if two MQC_Scalar variables are not equal*
- logical function [mqc_algebra::mqc_scalareq](#) (Scalar1, Scalar2)
***MQC_ScalarEQ** is a function that returns TRUE if two MQC_Scalar variables are equal*
- logical function [mqc_algebra::mqc_scalarlt](#) (Scalar1, Scalar2)
***MQC_ScalarLT** is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar*
- logical function [mqc_algebra::mqc_realltscalar](#) (RealIn, Scalar)
***MQC_RealLTScalar** is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar*
- logical function [mqc_algebra::mqc_scalarltreal](#) (Scalar, RealIn)
***MQC_ScalarLTReal** is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real*
- logical function [mqc_algebra::mqc_scalargt](#) (Scalar1, Scalar2)
***MQC_ScalarGT** is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar*
- logical function [mqc_algebra::mqc_integertgtscalar](#) (IntIn, Scalar)
***MQC_IntegerGTScalar** is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar*
- logical function [mqc_algebra::mqc_scalargtinteger](#) (Scalar, IntIn)
***MQC_ScalarGTInteger** is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer*
- logical function [mqc_algebra::mqc_realgtscalar](#) (RealIn, Scalar)
***MQC_RealGTScalar** is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar*
- logical function [mqc_algebra::mqc_scalargtreal](#) (Scalar, RealIn)
***MQC_ScalarGTReal** is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real*
- logical function [mqc_algebra::mqc_scalarle](#) (Scalar1, Scalar2)
***MQC_ScalarLE** is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar*
- logical function [mqc_algebra::mqc_reallescalar](#) (RealIn, Scalar)
***MQC_RealLEScalar** is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar*

- logical function [mqc_algebra::mqc_scalarlereal](#) (Scalar, RealIn)
***MQC_ScalarLEReal** is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real*
- logical function [mqc_algebra::mqc_integerlescalar](#) (IntIn, Scalar)
***MQC_IntegerLEScalar** is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar*
- logical function [mqc_algebra::mqc_scalarleinteger](#) (Scalar, IntIn)
***MQC_ScalarLEInteger** is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer*
- logical function [mqc_algebra::mqc_scalarge](#) (Scalar1, Scalar2)
***MQC_ScalarGE** is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_conjugate](#) (ScalarIn)
***MQC_Scalar_Complex_Conjugate** is a function that returns the complex conjugate of an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_realpart](#) (ScalarIn)
***MQC_Scalar_Complex_RealPart** is a function that returns the real part of an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_imagpart](#) (ScalarIn)
***MQC_Scalar_Complex_ImagPart** is a function that returns the inaginary part of an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
***MQC_IntegerScalarMultiply** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerMultiply** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_realscalarmultiply](#) (RealIn, Scalar)
***MQC_RealScalarMultiply** is a function that is used to multiply an intrinsic real by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealmultiply](#) (Scalar, RealIn)
***MQC_ScalarRealMultiply** is a function that is used to multiply an intrinsic real by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
***MQC_ComplexScalarMultiply** is a function that is used to multiply an intrinsic complex by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
***MQC_ScalarComplexMultiply** is a function that is used to multiply an intrinsic complex by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalardivide](#) (IntegerIn, Scalar)
***MQC_IntegerScalarDivide** is a function that is used to divide an intrinsic integer by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegerdivide](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerDivide** is a function that is used to divide an MQC_Scalar by an intrinsic integer*
- type(mqc_scalar) function [mqc_algebra::mqc_realscalardivide](#) (RealIn, Scalar)
***MQC_RealScalarDivide** is a function that is used to divide an intrinsic real by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealddivide](#) (Scalar, RealIn)
***MQC_ScalarRealDivide** is a function that is used to divide an MQC_Scalar by an intrinsic real*
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalardivide](#) (ComplexIn, Scalar)
***MQC_ComplexScalarDivide** is a function that is used to divide an intrinsic complex by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexdivide](#) (Scalar, ComplexIn)
***MQC_ScalarComplexDivide** is a function that is used to divide an MQC_Scalar by an intrinsic complex*
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalaradd](#) (IntegerIn, Scalar)
***MQC_IntegerScalarAdd** is a function that is used to multiply an intrinsic integer by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegeradd](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerAdd** is a function that is used to sum an intrinsic integer by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_realscalaradd](#) (RealIn, Scalar)
***MQC_RealScalarAdd** is a function that is used to sum an intrinsic real by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealadd](#) (Scalar, RealIn)
***MQC_ScalarRealAdd** is a function that is used to sum an intrinsic real by an MQC_Scalar*

- type(mqc_scalar) function [mqc_algebra::mqc_complexscalaradd](#) (ComplexIn, Scalar)
***MQC_ComplexScalarAdd** is a function that is used to sum an intrinsic complex by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexadd](#) (Scalar, ComplexIn)
***MQC_ScalarComplexAdd** is a function that is used to sum an intrinsic complex by an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalarsubtract](#) (IntegerIn, Scalar)
***MQC_IntegerScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic integer*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegersubtract](#) (Scalar, IntegerIn)
***MQC_ScalarIntegerSubtract** is a function that is used to subtract an intrinsic integer from an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_realscalarsubtract](#) (RealIn, Scalar)
***MQC_RealScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic real*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealsubtract](#) (Scalar, RealIn)
***MQC_ScalarRealSubtract** is a function that is used to subtract an intrinsic real from an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalarsubtract](#) (ComplexIn, Scalar)
***MQC_ComplexScalarSubtract** is a function that is used to subtract an MQC_Scalar from an intrinsic complex*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexsubtract](#) (Scalar, ComplexIn)
***MQC_ScalarComplexSubtract** is a function that is used to subtract an intrinsic complex from an MQC_Scalar*
- subroutine [mqc_algebra::mqc_allocate_vector](#) (N, Vector, Data_Type)
- subroutine [mqc_algebra::mqc_deallocate_vector](#) (Vector)
- integer(kind=int64) function [mqc_algebra::mqc_length_vector](#) (Vector)
- logical function [mqc_algebra::mqc_vector_havereal](#) (Vector)
- logical function [mqc_algebra::mqc_vector_haveinteger](#) (Vector)
- logical function [mqc_algebra::mqc_vector_havecomplex](#) (Vector)
- logical function [mqc_algebra::mqc_vector_iscolumn](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_int2real](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_int2complex](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_real2int](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_real2complex](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_complex2int](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_complex2real](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_scalar_at](#) (Vec, I)
- type(mqc_vector) function [mqc_algebra::mqc_vector_vector_at](#) (Vec, I, J)
- subroutine [mqc_algebra::mqc_set_vector2integerarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_vector2realarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_vector2complexarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_vector2vector](#) (VectorOut, VectorIn)
- type(mqc_vector) function [mqc_algebra::mqc_vectorvectorsum](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_vectorvectordifference](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_scalarvectorsum](#) (ScalarIn, VectorIn)
- type(mqc_vector) function [mqc_algebra::mqc_scalarvectordifference](#) (ScalarIn, VectorIn)
- type(mqc_vector) function [mqc_algebra::mqc_elementvectorproduct](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_vector_transpose](#) (Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vector_conjugate_transpose](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vectorvectordotproduct](#) (Vector1, Vector2)
- type(mqc_matrix) function [mqc_algebra::mqc_outer](#) (VA, VB)
- type(mqc_vector) function [mqc_algebra::mqc_crossproduct](#) (Vector1In, Vector2In)

- subroutine `mqc_algebra::mqc_print_vector_algebra1` (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- type(mqc_vector) function `mqc_algebra::mqc_vector_cast_real` (VA)
- type(mqc_vector) function `mqc_algebra::mqc_vector_cast_complex` (VA)
- subroutine `mqc_algebra::mqc_vector_scalar_put` (Vector, Scalar, I)
- subroutine `mqc_algebra::mqc_vector_scalar_increment` (Vector, Scalar, I)
- subroutine `mqc_algebra::mqc_vector_vector_put` (Vector, VectorIn, I)
- subroutine `mqc_algebra::mqc_vector_initialize` (Vector, Length, Scalar)
- type(mqc_vector) function `mqc_algebra::mqc_scalarvectorproduct` (Scalar, Vector)
- type(mqc_vector) function `mqc_algebra::mqc_vectorscalarproduct` (vector, scalar)
- type(mqc_vector) function `mqc_algebra::mqc_vectorscalardivide` (vector, scalar)
- type(mqc_vector) function `mqc_algebra::mqc_realvectorproduct` (Realln, Vector)
- type(mqc_vector) function `mqc_algebra::mqc_vectorrealproduct` (vector, realln)
- type(mqc_vector) function `mqc_algebra::mqc_vectorrealdive` (vector, realln)
- type(mqc_vector) function `mqc_algebra::mqc_integervectorproduct` (intln, Vector)
- type(mqc_vector) function `mqc_algebra::mqc_vectorintegerproduct` (vector, intln)
- type(mqc_vector) function `mqc_algebra::mqc_vectorintegerdivide` (vector, intln)
- type(mqc_vector) function `mqc_algebra::mqc_complexvectorproduct` (Compln, Vector)
- type(mqc_vector) function `mqc_algebra::mqc_vectorcomplexproduct` (vector, compln)
- type(mqc_vector) function `mqc_algebra::mqc_vectorcomplexdivide` (vector, compln)
- type(mqc_scalar) function `mqc_algebra::mqc_vector_norm` (vector, methodln)
- logical function `mqc_algebra::mqc_vector_isallocated` (Vector)
- subroutine `mqc_algebra::mqc_vector_push` (Vector, Scalar)
- subroutine `mqc_algebra::mqc_vector_unshift` (Vector, Scalar)
- type(mqc_scalar) function `mqc_algebra::mqc_vector_pop` (Vector)
- type(mqc_scalar) function `mqc_algebra::mqc_vector_shift` (Vector)
- type(mqc_scalar) function `mqc_algebra::mqc_vector_maxval` (Vector)
- type(mqc_scalar) function `mqc_algebra::mqc_vector_minval` (Vector)
- integer function `mqc_algebra::mqc_vector_maxloc` (Vector)
- integer function `mqc_algebra::mqc_vector_minloc` (Vector)
- type(mqc_vector) function `mqc_algebra::mqc_vector_argsort` (Vector)
- subroutine `mqc_algebra::mqc_vector_sort` (Vector, idx)
- subroutine `mqc_algebra::mqc_vector_sqrt` (A)
- type(mqc_vector) function `mqc_algebra::mqc_vector_abs` (A)
- subroutine `mqc_algebra::mqc_vector_power` (A, P)
- type(mqc_vector) function `mqc_algebra::mqc_vector_complex_realpart` (A)
- type(mqc_vector) function `mqc_algebra::mqc_vector_complex_imagpart` (A)
- type(mqc_vector) function `mqc_algebra::mqc_vector_cmplx` (Vector1, Vector2)
- character(len=64) function `mqc_algebra::mqc_matrix_storage_type` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_diagonalize` (A, EVals, EVecs)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_cast_real` (MA)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_cast_complex` (MA)
- type(mqc_scalar) function `mqc_algebra::mqc_matrix_scalar_at` (Mat, I, J)
- type(mqc_vector) function `mqc_algebra::mqc_matrix_vector_at` (Mat, Rows, Cols)
- recursive subroutine `mqc_algebra::mqc_matrix_vector_put` (Mat, VectorIn, Rows, Cols)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_matrix_at` (Mat, Rows, Cols)

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_vector` (diagVectorIn, mat)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_integer` (mat, diagMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_real` (mat, diagMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_complex` (mat, diagMatrixIn)

- subroutine [mqc_algebra::mqc_matrix_symmmatrix_put_integer](#) (mat, symmMatrixIn)
- subroutine [mqc_algebra::mqc_matrix_symmmatrix_put_real](#) (mat, symmMatrixIn)
- subroutine [mqc_algebra::mqc_matrix_symmmatrix_put_complex](#) (mat, symmMatrixIn)
- recursive subroutine [mqc_algebra::mqc_matrix_matrix_put](#) (Mat, MatrixIn, Rows, Cols)
- integer(kind=int64) function [mqc_algebra::symindexhash](#) (i, j, k, l)
- type(mqc_matrix) function [mqc_algebra::mqc_elementmatrixproduct](#) (A, B)
- type(mqc_matrix) function [mqc_algebra::mqc_elementmatrixdivide](#) (A, B)
- logical function [mqc_algebra::mqc_matrix_test_symmetric](#) (Matrix, Option)
- logical function [mqc_algebra::mqc_matrix_test_diagonal](#) (Matrix)
- subroutine [mqc_algebra::mqc_allocate_matrix](#) (M, N, Matrix, Data_Type, Storage)
- subroutine [mqc_algebra::mqc_deallocate_matrix](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_isallocated](#) (Matrix)
- subroutine [mqc_algebra::mqc_set_integerarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_realarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_complexarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_matrix2integerarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_algebra::mqc_set_matrix2realarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_algebra::mqc_set_matrix2complexarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_algebra::mqc_set_matrix2matrix](#) (MatrixOut, MatrixIn)
- subroutine [mqc_algebra::mqc_print_matrix_algebra1](#) (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_algebra::mqc_matrix_copy_int2real](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_copy_int2complex](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_copy_real2int](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_copy_real2complex](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_copy_complex2int](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_copy_complex2real](#) (Matrix)
- integer(kind=int64) function [mqc_algebra::mqc_matrix_rows](#) (Matrix)
- integer(kind=int64) function [mqc_algebra::mqc_matrix_columns](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_havereal](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_haveinteger](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_havecomplex](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_havefull](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_havesymmetric](#) (Matrix)
- logical function [mqc_algebra::mqc_matrix_havediagonal](#) (Matrix)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_transpose](#) (Matrix)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_conjugate_transpose](#) (Matrix)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_symmetrize](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_full2symm](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_symm2full](#) (Matrix, Option)
- subroutine [mqc_algebra::mqc_matrix_full2diag](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_diag2full](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_symm2diag](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_diag2symm](#) (Matrix)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_symm2full_func](#) (Matrix)
- subroutine [mqc_algebra::matrix_symm2sq_integer](#) (N, I_Symm, I_Sq)
- subroutine [mqc_algebra::matrix_symm2sq_real](#) (N, A_Symm, A_Sq)
- subroutine [mqc_algebra::matrix_symm2sq_complex](#) (N, A_Symm, A_Sq)
- type(mqc_matrix) function [mqc_algebra::mqc_vector2diagmatrix](#) (vector)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixsum](#) (MA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixsubtract](#) (MA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixproduct](#) (MA, MB)

- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixdotproduct](#) (MA, MB)
- type(mqc_vector) function [mqc_algebra::mqc_matrixvectordotproduct](#) (MA, VB)
- type(mqc_vector) function [mqc_algebra::mqc_vectormatrixdotproduct](#) (VA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type(mqc_matrix) function [mqc_algebra::mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_matrix_contraction](#) (Matrix1, Matrix2)
- subroutine [mqc_algebra::mqc_matrix_scalar_put](#) (Matrix, Scalar, I, J)
- subroutine [mqc_algebra::mqc_matrix_initialize](#) (Matrix, Rows, Columns, Scalar, Storage)
- subroutine [mqc_algebra::mqc_matrix_identity](#) (matrix, n, m)
- subroutine [mqc_algebra::mqc_matrix_set](#) (matrix, scalar, storage)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_norm](#) (matrix, methodIn)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_determinant](#) (a)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_inverse](#) (a)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_trace](#) (matrix)
- subroutine [mqc_algebra::mqc_matrix_generalized_eigensystem](#) (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine [mqc_algebra::mqc_matrix_svd](#) (A, EVals, EUVecs, EVVecs)
- subroutine [mqc_algebra::mqc_matrix_rms_max](#) (A, rms_A, max_A)
- subroutine [mqc_algebra::mqc_matrix_sqrt](#) (A, eVals, eVecs)
- type(mqc_matrix) function [mqc_algebra::mqc_givens_matrix](#) (m_size, angle, p, q)
- subroutine [mqc_algebra::mqc_allocate_r4tensor](#) (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine [mqc_algebra::mqc_deallocate_r4tensor](#) (Tensor)
- type(mqc_scalar) function [mqc_algebra::mqc_r4tensor_at](#) (Tensor, I, J, K, L)
- subroutine [mqc_algebra::mqc_r4tensor_put](#) (Tensor, Element, I, J, K, L)
- subroutine [mqc_algebra::mqc_print_r4tensor_algebra1](#) (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine [mqc_algebra::mqc_set_array2tensor](#) (TensorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_r4tensor_initialize](#) (R4Tensor, I, J, K, L, Scalar)
- subroutine [mqc_algebra::mqc_matrix_symmsymmr4tensor_put_real](#) (r4Tensor, symmSymmMatrixIn)
- subroutine [mqc_algebra::mqc_matrix_symmsymmr4tensor_put_complex](#) (r4Tensor, symmSymmMatrixIn)
- logical function [mqc_algebra::mqc_r4tensor_haveinteger](#) (R4Tensor)
- logical function [mqc_algebra::mqc_r4tensor_havereal](#) (R4Tensor)
- logical function [mqc_algebra::mqc_r4tensor_havecomplex](#) (R4Tensor)

7.2 src/mqc_est.F03 File Reference

Data Types

- type [mqc_est::mqc_scf_integral](#)
- type [mqc_est::mqc_scf_eigenvalues](#)
- type [mqc_est::mqc_wavefunction](#)
- type [mqc_est::mqc_pscf_wavefunction](#)
- type [mqc_est::mqc_determinant_string](#)
- type [mqc_est::mqc_determinant](#)
- type [mqc_est::mqc_twoeris](#)
- interface [mqc_est::mqc_print](#)
- interface [mqc_est::matmul](#)
- interface [mqc_est::dot_product](#)
- interface [mqc_est::transpose](#)
- interface [mqc_est::dagger](#)

- interface [mqc_est::contraction](#)
- interface [mqc_est::mqc_matrix_undospinblockghf](#)
- interface [mqc_est::assignment\(=\)](#)
- interface [mqc_est::operator\(+\)](#)
- interface [mqc_est::operator\(-\)](#)
- interface [mqc_est::operator\(*\)](#)

Modules

- module [mqc_est](#)

Functions/Subroutines

- subroutine [mqc_est::mqc_print_wavefunction](#) (wavefunction, iOut, label)
- subroutine [mqc_est::mqc_print_integral](#) (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_est::mqc_print_eigenvalues](#) (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_est::mqc_print_twoeris](#) (twoERIs, iOut, header, blank_at_top, blank_at_bottom)
- logical function [mqc_est::mqc_integral_isallocated](#) (Integral)
- logical function [mqc_est::mqc_eigenvalues_isallocated](#) (Eigenvalues)
- logical function [mqc_est::mqc_integral_has_alpha](#) (integral)
- logical function [mqc_est::mqc_integral_has_beta](#) (integral)
- logical function [mqc_est::mqc_integral_has_alphabeta](#) (integral)
- logical function [mqc_est::mqc_integral_has_betaalpha](#) (integral)
- logical function [mqc_est::mqc_eigenvalues_has_alpha](#) (eigenvalues)
- logical function [mqc_est::mqc_eigenvalues_has_beta](#) (eigenvalues)
- character(len=64) function [mqc_est::mqc_integral_array_type](#) (integral)
- character(len=64) function [mqc_est::mqc_eigenvalues_array_type](#) (eigenvalues)
- character(len=64) function [mqc_est::mqc_integral_array_name](#) (integral)
- character(len=64) function [mqc_est::mqc_eigenvalues_array_name](#) (eigenvalues)
- subroutine [mqc_est::mqc_integral_add_name](#) (integral, arrayName)
- subroutine [mqc_est::mqc_eigenvalues_add_name](#) (eigenvalues, arrayName)
- integer(kind=int64) function [mqc_est::mqc_integral_dimension](#) (integral, label, axis)
- integer(kind=int64) function [mqc_est::mqc_eigenvalues_dimension](#) (eigenvalues, label)
- subroutine [mqc_est::mqc_twoeris_allocate](#) (twoERIs, storageType, integralType, alpha, beta, alphaBeta, beta↔Alpha)
- subroutine [mqc_est::mqc_integral_allocate](#) (integral, arrayName, arrayType, alpha, beta, alphaBeta, betaAlpha)
- subroutine [mqc_est::mqc_eigenvalues_allocate](#) (eigenvalues, arrayName, arrayType, alpha, beta)
- subroutine [mqc_est::mqc_integral_identity](#) (integral, nAlpha, nBeta, label, nAlpha2, nBeta2)
- subroutine [mqc_est::mqc_integral_initialize](#) (integral, nAlpha, nBeta, scalar, label, nAlpha2, nBeta2)
- type(mqc_matrix) function [mqc_est::mqc_integral_output_block](#) (integral, blockName)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_output_orbitals](#) (integral, orbString, alphaOrbsIn, beta↔OrbsIn, axis)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_swap_orbitals](#) (integral, alphaOrbsIn, betaOrbsIn, axis)
- type(mqc_vector) function [mqc_est::mqc_eigenvalues_output_block](#) (eigenvalues, blockName)
- subroutine [mqc_est::mqc_integral_output_array](#) (matrixOut, integralln)
- subroutine [mqc_est::mqc_eigenvalues_output_array](#) (vectorOut, eigenvaluesIn)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_matrix_multiply](#) (integralA, matrixB, label)
- type(mqc_scf_integral) function [mqc_est::mqc_matrix_integral_multiply](#) (matrixA, integralB, label)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_sum](#) (integralA, integralB)

- type(mqc_scf_integral) function [mqc_est::mqc_integral_difference](#) (integralA, integralB)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_integral_multiply](#) (integralA, integralB, label)
- type(mqc_scf_integral) function [mqc_est::mqc_scalar_integral_multiply](#) (scalar, integral)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_scalar_multiply](#) (integral, scalar)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_eigenvalues_multiply](#) (integralA, eigenvaluesB, label)
- type(mqc_scf_integral) function [mqc_est::mqc_eigenvalues_integral_multiply](#) (eigenvaluesA, integralB, label)
- type(mqc_scf_eigenvalues) function [mqc_est::mqc_eigenvalues_eigenvalues_multiply](#) (eigenvaluesA, eigenvaluesB, label)
- type(mqc_scalar) function [mqc_est::mqc_eigenvalue_eigenvalue_dotproduct](#) (eigenvalueA, eigenvalueB)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_transpose](#) (integral, label)
- type(mqc_scf_integral) function [mqc_est::mqc_integral_conjugate_transpose](#) (integral, label)
- type(mqc_scalar) function [mqc_est::mqc_integral_norm](#) (integral, methodIn)
- subroutine [mqc_est::mqc_matrix_spinblockghf](#) (array, nelec, multi, elist)
- subroutine [mqc_est::mqc_matrix_undospinblockghf_eigenvalues](#) (eigenvaluesIn, vectorOut)
- subroutine [mqc_est::mqc_matrix_undospinblockghf_integral](#) (integralIn, matrixOut)
- type(mqc_scalar) function [mqc_est::mqc_scf_integral_contraction](#) (integral1, integral2)
- type(mqc_scf_integral) function [mqc_est::mqc_eri_integral_contraction](#) (eris, integral, label)
- subroutine [mqc_est::mqc_scf_integral_generalized_eigensystem](#) (integralA, integralB, eVals, rEVecs, IEVecs)
- subroutine [mqc_est::mqc_scf_integral_diagonalize](#) (integral, eVals, eVecs)
- type(mqc_scf_integral) function [mqc_est::mqc_scf_integral_inverse](#) (integral)
- type(mqc_scalar) function [mqc_est::mqc_scf_integral_trace](#) (integral)
- type(mqc_scalar) function [mqc_est::mqc_scf_integral_determinant](#) (integral)
- subroutine [mqc_est::mqc_integral_set_energy_list](#) (integral, elist)
- integer(kind=int64) function, dimension(:), allocatable [mqc_est::mqc_integral_get_energy_list](#) (integral)
- subroutine [mqc_est::mqc_integral_delete_energy_list](#) (integral)
- subroutine [mqc_est::mqc_scf_eigenvalues_power](#) (eigenvalues, power)
- type(mqc_scalar) function [mqc_est::mqc_twoeris_at](#) (twoERIs, i, j, k, l, spinBlock)
- type(mqc_scalar) function [mqc_est::mqc_integral_at](#) (integral, i, j, spinBlock)
- type(mqc_scalar) function [mqc_est::mqc_eigenvalues_at](#) (eigenvalues, i, spinBlock)
- subroutine [mqc_est::mqc_scf_transformation_matrix](#) (overlap, transform_matrix, nBasUse)
- subroutine [mqc_est::gen_det_str](#) (IOut, IPrint, NBasisIn, NAlphaIn, NBetaIn, Determinants, NCoreIn)
- type(mqc_scalar) function [mqc_est::slater_condon](#) (IOut, IPrint, NBasisIn, Determinants, L_A_String, L_B_String, R_A_String, R_B_String, Core_Hamiltonian, ERIs, UHF)
- subroutine [mqc_est::twoeri_trans](#) (IOut, IPrint, MO_Coeff, ERIs, MO_ERIs, UHF)
- subroutine [mqc_est::mqc_build_ci_hamiltonian](#) (IOut, IPrint, NBasis, Determinants, MO_Core_Ham, MO_ERIs, UHF, CI_Hamiltonian)
- type(mqc_matrix) function [mqc_est::get_one_gamma_matrix](#) (iOut, iPrint, nBasisIn, nState, determinants, ci_← amplitudes, nCoreIn, nOrbsIn)

Index

- abs
 - mqc_algebra::mqc_scalar, [161](#)
 - mqc_algebra::mqc_vector, [169](#)
- addlabel
 - mqc_est::mqc_scf_eigenvalues, [163](#)
 - mqc_est::mqc_scf_integral, [164](#)
- alpha
 - mqc_est::mqc_determinant_string, [146](#)
- argsort
 - mqc_algebra::mqc_vector, [169](#)
- at
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_algebra::mqc_r4tensor, [160](#)
 - mqc_algebra::mqc_vector, [169](#)
 - mqc_est::mqc_scf_eigenvalues, [163](#)
- basis
 - mqc_est::mqc_wavefunction, [174](#)
- beta
 - mqc_est::mqc_determinant_string, [146](#)
- bin_coeff
 - mqc_algebra, [17](#)
- charge
 - mqc_est::mqc_wavefunction, [174](#)
- core_hamiltonian
 - mqc_est::mqc_wavefunction, [174](#)
- cval
 - mqc_algebra::mqc_scalar, [162](#)
- dagger
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_algebra::mqc_vector, [169](#)
- data_type
 - mqc_algebra::mqc_vector, [172](#)
- deleteelist
 - mqc_est::mqc_scf_integral, [164](#)
- density_matrix
 - mqc_est::mqc_wavefunction, [175](#)
- det
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_est::mqc_scf_integral, [164](#)
- diag
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_algebra::mqc_vector, [169](#)
 - mqc_est::mqc_scf_integral, [165](#)
- eigensys
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_est::mqc_scf_integral, [165](#)
- factorial
 - mqc_algebra, [18](#)
- fock_matrix
 - mqc_est::mqc_wavefunction, [175](#)
- gen_det_str
 - mqc_est, [104](#)
- get_one_gamma_matrix
 - mqc_est, [104](#)
- getblock
 - mqc_est::mqc_scf_eigenvalues, [163](#)
 - mqc_est::mqc_scf_integral, [165](#)
- getelist
 - mqc_est::mqc_scf_integral, [165](#)
- getlabel
 - mqc_est::mqc_scf_eigenvalues, [163](#)
 - mqc_est::mqc_scf_integral, [165](#)
- identity
 - mqc_algebra::mqc_matrix, [149](#)
 - mqc_est::mqc_scf_integral, [165](#)
- init
 - mqc_algebra::mqc_matrix, [150](#)
 - mqc_algebra::mqc_r4tensor, [160](#)
 - mqc_algebra::mqc_vector, [169](#)
 - mqc_est::mqc_scf_integral, [165](#)
- initialize
 - mqc_algebra::mqc_matrix, [150](#)
 - mqc_algebra::mqc_r4tensor, [161](#)
 - mqc_algebra::mqc_vector, [170](#)
- inv
 - mqc_algebra::mqc_matrix, [150](#)
 - mqc_est::mqc_scf_integral, [166](#)
- ival
 - mqc_algebra::mqc_scalar, [162](#)
- length
 - mqc_algebra::mqc_vector, [172](#)
- mat
 - mqc_algebra::mqc_matrix, [150](#)
- matc

- mqc_algebra::mqc_matrix, 152
- mati
 - mqc_algebra::mqc_matrix, 152
- matr
 - mqc_algebra::mqc_matrix, 152
- matrix_symm2sq_complex
 - mqc_algebra, 18
 - mqc_algebra::matrix_symm2sq, 142
- matrix_symm2sq_integer
 - mqc_algebra, 18
 - mqc_algebra::matrix_symm2sq, 143
- matrix_symm2sq_real
 - mqc_algebra, 19
 - mqc_algebra::matrix_symm2sq, 143
- maxloc
 - mqc_algebra::mqc_vector, 170
- maxval
 - mqc_algebra::mqc_vector, 170
- minloc
 - mqc_algebra::mqc_vector, 170
- minval
 - mqc_algebra::mqc_vector, 170
- mo_coefficients
 - mqc_est::mqc_wavefunction, 175
- mo_energies
 - mqc_est::mqc_wavefunction, 175
- mo_symmetries
 - mqc_est::mqc_wavefunction, 175
- mput
 - mqc_algebra::mqc_matrix, 150
- mqc_algebra, 9
 - bin_coeff, 17
 - factorial, 18
 - matrix_symm2sq_complex, 18
 - matrix_symm2sq_integer, 18
 - matrix_symm2sq_real, 19
 - mqc_allocate_matrix, 19
 - mqc_allocate_r4tensor, 19
 - mqc_allocate_scalar, 19
 - mqc_allocate_vector, 20
 - mqc_complexscalaradd, 20
 - mqc_complexscalardivide, 21
 - mqc_complexscalarmultiply, 22
 - mqc_complexscalarsubtract, 23
 - mqc_complexvectorproduct, 23
 - mqc_crossproduct, 23
 - mqc_deallocate_matrix, 24
 - mqc_deallocate_r4tensor, 24
 - mqc_deallocate_scalar, 24
 - mqc_deallocate_vector, 25
 - mqc_elementmatrixdivide, 25
 - mqc_elementmatrixproduct, 25
 - mqc_elementvectorproduct, 25
 - mqc_givens_matrix, 25
 - mqc_input_complex_scalar, 25
 - mqc_input_integer_scalar, 26
 - mqc_input_real_scalar, 27
 - mqc_integertscalar, 28
 - mqc_integerlescalar, 28
 - mqc_integerscalaradd, 29
 - mqc_integerscalardivide, 30
 - mqc_integerscalarmultiply, 30
 - mqc_integerscalarsubtract, 31
 - mqc_integervectorproduct, 32
 - mqc_length_vector, 32
 - mqc_matrix_cast_complex, 32
 - mqc_matrix_cast_real, 32
 - mqc_matrix_columns, 32
 - mqc_matrix_conjugate_transpose, 32
 - mqc_matrix_copy_complex2int, 33
 - mqc_matrix_copy_complex2real, 33
 - mqc_matrix_copy_int2complex, 33
 - mqc_matrix_copy_int2real, 33
 - mqc_matrix_copy_real2complex, 33
 - mqc_matrix_copy_real2int, 33
 - mqc_matrix_determinant, 34
 - mqc_matrix_diag2full, 34
 - mqc_matrix_diag2symm, 34
 - mqc_matrix_diagmatrix_put_complex, 34
 - mqc_matrix_diagmatrix_put_integer, 34
 - mqc_matrix_diagmatrix_put_real, 34
 - mqc_matrix_diagmatrix_put_vector, 35
 - mqc_matrix_diagonalize, 35
 - mqc_matrix_full2diag, 35
 - mqc_matrix_full2symm, 35
 - mqc_matrix_generalized_eigensystem, 35
 - mqc_matrix_havecomplex, 35
 - mqc_matrix_havediagonal, 36
 - mqc_matrix_havetfull, 36
 - mqc_matrix_haveinteger, 36
 - mqc_matrix_havereal, 36
 - mqc_matrix_havesymmetric, 36
 - mqc_matrix_identity, 36
 - mqc_matrix_initialize, 37
 - mqc_matrix_inverse, 37
 - mqc_matrix_isallocated, 37
 - mqc_matrix_matrix_at, 37
 - mqc_matrix_matrix_contraction, 38
 - mqc_matrix_matrix_put, 38
 - mqc_matrix_norm, 38
 - mqc_matrix_rms_max, 39
 - mqc_matrix_rows, 39
 - mqc_matrix_scalar_at, 39
 - mqc_matrix_scalar_put, 39
 - mqc_matrix_set, 39
 - mqc_matrix_sqrt, 40
 - mqc_matrix_storage_type, 40
 - mqc_matrix_svd, 40

[mqc_matrix_symm2diag](#), 40
[mqc_matrix_symm2full](#), 40
[mqc_matrix_symm2full_func](#), 41
[mqc_matrix_symmetrize](#), 41
[mqc_matrix_symmmatrix_put_complex](#), 41
[mqc_matrix_symmmatrix_put_integer](#), 41
[mqc_matrix_symmmatrix_put_real](#), 41
[mqc_matrix_symmsymmr4tensor_put_complex](#), 41
[mqc_matrix_symmsymmr4tensor_put_real](#), 42
[mqc_matrix_test_diagonal](#), 42
[mqc_matrix_test_symmetric](#), 42
[mqc_matrix_trace](#), 42
[mqc_matrix_transpose](#), 42
[mqc_matrix_vector_at](#), 42
[mqc_matrix_vector_put](#), 43
[mqc_matrixmatrixdotproduct](#), 43
[mqc_matrixmatrixproduct](#), 43
[mqc_matrixmatrixsubtract](#), 43
[mqc_matrixmatrixsum](#), 43
[mqc_matrixscalarproduct](#), 44
[mqc_matrixvectordotproduct](#), 44
[mqc_outer](#), 44
[mqc_output_complex_scalar](#), 44
[mqc_output_integer_scalar](#), 45
[mqc_output_mqcscalar_scalar](#), 46
[mqc_output_real_scalar](#), 46
[mqc_print_matrix_algebra1](#), 47
[mqc_print_r4tensor_algebra1](#), 47
[mqc_print_scalar_algebra1](#), 47
[mqc_print_vector_algebra1](#), 48
[mqc_r4tensor_at](#), 49
[mqc_r4tensor_havecomplex](#), 49
[mqc_r4tensor_haveinteger](#), 49
[mqc_r4tensor_havereal](#), 49
[mqc_r4tensor_initialize](#), 49
[mqc_r4tensor_put](#), 50
[mqc_realgtscalar](#), 50
[mqc_realltscalar](#), 51
[mqc_realscalaradd](#), 52
[mqc_realscalardivide](#), 53
[mqc_realscalarmultiply](#), 54
[mqc_realscalarsubtract](#), 54
[mqc_realvectorproduct](#), 55
[mqc_scalar_acos](#), 55
[mqc_scalar_asin](#), 56
[mqc_scalar_atan](#), 56
[mqc_scalar_atan2](#), 57
[mqc_scalar_cmplx](#), 57
[mqc_scalar_complex_conjugate](#), 58
[mqc_scalar_complex_imagpart](#), 59
[mqc_scalar_complex_realpart](#), 59
[mqc_scalar_cos](#), 60
[mqc_scalar_get_abs_value](#), 61
[mqc_scalar_get_intrinsic_complex](#), 61
[mqc_scalar_get_intrinsic_integer](#), 62
[mqc_scalar_get_intrinsic_real](#), 62
[mqc_scalar_get_random_value](#), 63
[mqc_scalar_havecomplex](#), 64
[mqc_scalar_haveinteger](#), 64
[mqc_scalar_havereal](#), 65
[mqc_scalar_isallocated](#), 66
[mqc_scalar_sin](#), 66
[mqc_scalar_sqrt](#), 67
[mqc_scalar_tan](#), 67
[mqc_scalaradd](#), 68
[mqc_scalarcomplexadd](#), 69
[mqc_scalarcomplexdivide](#), 69
[mqc_scalarcomplexexponent](#), 70
[mqc_scalarcomplexmultiply](#), 71
[mqc_scalarcomplexsubtract](#), 71
[mqc_scalardivide](#), 72
[mqc_scalareq](#), 73
[mqc_scalarexponent](#), 73
[mqc_scalarge](#), 74
[mqc_scalargt](#), 75
[mqc_scalargtinteger](#), 75
[mqc_scalargtreal](#), 76
[mqc_scalarintegeradd](#), 77
[mqc_scalarintegerdivide](#), 78
[mqc_scalarintegerexponent](#), 78
[mqc_scalarintegermultiply](#), 79
[mqc_scalarintegersubtract](#), 80
[mqc_scalarle](#), 80
[mqc_scalarleinteger](#), 81
[mqc_scalarlereal](#), 82
[mqc_scalarlt](#), 83
[mqc_scalarltreal](#), 83
[mqc_scalarmatrixproduct](#), 84
[mqc_scalarmultiply](#), 84
[mqc_scalarne](#), 85
[mqc_scalarrealadd](#), 86
[mqc_scalarrealdivide](#), 87
[mqc_scalarrealexponent](#), 87
[mqc_scalarrealmultiply](#), 88
[mqc_scalarrealsubtract](#), 89
[mqc_scalarsubtract](#), 89
[mqc_scalarvectordifference](#), 90
[mqc_scalarvectorproduct](#), 90
[mqc_scalarvectorsum](#), 90
[mqc_set_array2tensor](#), 90
[mqc_set_array2vector_complex](#), 91
[mqc_set_array2vector_integer](#), 91
[mqc_set_array2vector_real](#), 91
[mqc_set_complexarray2matrix](#), 91
[mqc_set_integerarray2matrix](#), 91
[mqc_set_matrix2complexarray](#), 91
[mqc_set_matrix2integerarray](#), 92

- mqc_set_matrix2matrix, 92
- mqc_set_matrix2realarray, 92
- mqc_set_realarray2matrix, 92
- mqc_set_vector2complexarray, 92
- mqc_set_vector2integerarray, 92
- mqc_set_vector2realarray, 93
- mqc_set_vector2vector, 93
- mqc_vector2diagmatrix, 93
- mqc_vector_abs, 93
- mqc_vector_argsort, 93
- mqc_vector_cast_complex, 93
- mqc_vector_cast_real, 94
- mqc_vector_cmplx, 94
- mqc_vector_complex_imagpart, 94
- mqc_vector_complex_realpart, 94
- mqc_vector_conjugate_transpose, 94
- mqc_vector_copy_complex2int, 94
- mqc_vector_copy_complex2real, 95
- mqc_vector_copy_int2complex, 95
- mqc_vector_copy_int2real, 95
- mqc_vector_copy_real2complex, 95
- mqc_vector_copy_real2int, 95
- mqc_vector_havecomplex, 95
- mqc_vector_haveinteger, 96
- mqc_vector_havereal, 96
- mqc_vector_initialize, 96
- mqc_vector_isallocated, 96
- mqc_vector_iscolumn, 96
- mqc_vector_maxloc, 96
- mqc_vector_maxval, 97
- mqc_vector_minloc, 97
- mqc_vector_minval, 97
- mqc_vector_norm, 97
- mqc_vector_pop, 97
- mqc_vector_power, 97
- mqc_vector_push, 98
- mqc_vector_scalar_at, 98
- mqc_vector_scalar_increment, 98
- mqc_vector_scalar_put, 98
- mqc_vector_shift, 98
- mqc_vector_sort, 98
- mqc_vector_sqrt, 99
- mqc_vector_transpose, 99
- mqc_vector_unshift, 99
- mqc_vector_vector_at, 99
- mqc_vector_vector_put, 99
- mqc_vectorcomplexdivide, 99
- mqc_vectorcomplexproduct, 100
- mqc_vectorintegerdivide, 100
- mqc_vectorintegerproduct, 100
- mqc_vectormatrixdotproduct, 100
- mqc_vectorrealdive, 100
- mqc_vectorrealproduct, 100
- mqc_vectorscalardivide, 101
- mqc_vectorscalarproduct, 101
- mqc_vectorvectordifference, 101
- mqc_vectorvectordotproduct, 101
- mqc_vectorvectorsum, 101
- symindexhash, 101
- mqc_algebra::abs, 119
 - mqc_scalar_get_abs_value, 119
 - mqc_vector_abs, 120
- mqc_algebra::acos, 120
 - mqc_scalar_acos, 120
- mqc_algebra::aimag, 121
 - mqc_scalar_complex_imagpart, 121
 - mqc_vector_complex_imagpart, 122
- mqc_algebra::asin, 122
 - mqc_scalar_asin, 122
- mqc_algebra::assignment(=), 123
 - mqc_input_complex_scalar, 124
 - mqc_input_integer_scalar, 124
 - mqc_input_real_scalar, 125
 - mqc_output_complex_scalar, 126
 - mqc_output_integer_scalar, 126
 - mqc_output_mqcscalar_scalar, 127
 - mqc_output_real_scalar, 128
 - mqc_set_array2tensor, 128
 - mqc_set_array2vector_complex, 128
 - mqc_set_array2vector_integer, 129
 - mqc_set_array2vector_real, 129
 - mqc_set_complexarray2matrix, 129
 - mqc_set_integerarray2matrix, 129
 - mqc_set_matrix2complexarray, 129
 - mqc_set_matrix2integerarray, 129
 - mqc_set_matrix2matrix, 130
 - mqc_set_matrix2realarray, 130
 - mqc_set_realarray2matrix, 130
 - mqc_set_vector2complexarray, 130
 - mqc_set_vector2integerarray, 130
 - mqc_set_vector2realarray, 130
 - mqc_set_vector2vector, 131
- mqc_algebra::atan, 132
 - mqc_scalar_atan, 132
- mqc_algebra::atan2, 132
 - mqc_scalar_atan2, 133
- mqc_algebra::cmplx, 133
 - mqc_scalar_cmplx, 134
 - mqc_vector_cmplx, 134
- mqc_algebra::conjg, 135
 - mqc_scalar_complex_conjugate, 135
- mqc_algebra::contraction, 135
 - mqc_matrix_matrix_contraction, 136
- mqc_algebra::cos, 137
 - mqc_scalar_cos, 137
- mqc_algebra::dagger, 137
 - mqc_matrix_conjugate_transpose, 138
 - mqc_vector_conjugate_transpose, 138

- mqc_algebra::dot_product, 139
 - mqc_vectorvectordotproduct, 139
- mqc_algebra::matmul, 140
 - mqc_matrixmatrixdotproduct, 140
 - mqc_matrixvectordotproduct, 140
 - mqc_vectormatrixdotproduct, 140
- mqc_algebra::matrix_symm2sq, 142
 - matrix_symm2sq_complex, 142
 - matrix_symm2sq_integer, 143
 - matrix_symm2sq_real, 143
- mqc_algebra::mqc_cast_complex, 143
 - mqc_matrix_cast_complex, 143
 - mqc_vector_cast_complex, 143
- mqc_algebra::mqc_cast_real, 144
 - mqc_matrix_cast_real, 144
 - mqc_vector_cast_real, 144
- mqc_algebra::mqc_have_complex, 146
 - mqc_matrix_havecomplex, 146
 - mqc_vector_havecomplex, 146
- mqc_algebra::mqc_have_int, 147
 - mqc_matrix_haveinteger, 147
 - mqc_vector_haveinteger, 147
- mqc_algebra::mqc_have_real, 147
 - mqc_matrix_havereal, 147
 - mqc_vector_havereal, 148
- mqc_algebra::mqc_matrix, 148
 - at, 149
 - dagger, 149
 - det, 149
 - diag, 149
 - eigensys, 149
 - identity, 149
 - init, 150
 - initialize, 150
 - inv, 150
 - mat, 150
 - matc, 152
 - mati, 152
 - matr, 152
 - mput, 150
 - norm, 150
 - print, 150
 - put, 151
 - rmsmax, 151
 - s_type, 151
 - set, 151
 - sqrt, 151
 - svd, 151
 - trace, 151
 - transpose, 152
 - vat, 152
 - vput, 152
- mqc_algebra::mqc_matrix_diagmatrix_put, 153
 - mqc_matrix_diagmatrix_put_complex, 153
 - mqc_matrix_diagmatrix_put_integer, 153
 - mqc_matrix_diagmatrix_put_real, 153
 - mqc_matrix_diagmatrix_put_vector, 153
- mqc_algebra::mqc_matrix_symmmatrix_put, 154
 - mqc_matrix_symmmatrix_put_complex, 154
 - mqc_matrix_symmmatrix_put_integer, 154
 - mqc_matrix_symmmatrix_put_real, 154
- mqc_algebra::mqc_print, 155
 - mqc_print_matrix_algebra1, 155
 - mqc_print_r4tensor_algebra1, 156
 - mqc_print_scalar_algebra1, 156
 - mqc_print_vector_algebra1, 157
- mqc_algebra::mqc_r4tensor, 160
 - at, 160
 - init, 160
 - initialize, 161
 - print, 161
 - put, 161
- mqc_algebra::mqc_scalar, 161
 - abs, 161
 - cval, 162
 - ival, 162
 - print, 162
 - random, 162
 - rval, 162
- mqc_algebra::mqc_set_array2vector, 167
 - mqc_set_array2vector_complex, 167
 - mqc_set_array2vector_integer, 167
 - mqc_set_array2vector_real, 167
- mqc_algebra::mqc_vector, 168
 - abs, 169
 - argsort, 169
 - at, 169
 - dagger, 169
 - data_type, 172
 - diag, 169
 - init, 169
 - initialize, 170
 - length, 172
 - maxloc, 170
 - maxval, 170
 - minloc, 170
 - minval, 170
 - norm, 170
 - pop, 170
 - power, 171
 - print, 171
 - push, 171
 - put, 171
 - shift, 171
 - size, 171
 - sort, 171
 - sqrt, 172
 - transpose, 172

- unshift, 172
- vat, 172
- vecc, 173
- veci, 173
- vecr, 173
- vput, 172
- mqc_algebra::operator(**), 185
 - mqc_scalarcomplexexponent, 185
 - mqc_scalarexponent, 186
 - mqc_scalarintegerexponent, 187
 - mqc_scalarrealexponent, 187
- mqc_algebra::operator(*), 177
 - mqc_complexscalarmultiply, 178
 - mqc_complexvectorproduct, 178
 - mqc_integerscalarmultiply, 178
 - mqc_integervectorproduct, 179
 - mqc_matrixmatrixproduct, 179
 - mqc_matrixscalarproduct, 179
 - mqc_realscalarmultiply, 180
 - mqc_realvectorproduct, 180
 - mqc_scalarcomplexmultiply, 180
 - mqc_scalarintegermultiply, 181
 - mqc_scalarmatrixproduct, 182
 - mqc_scalarmultiply, 182
 - mqc_scalarrealmultiply, 183
 - mqc_scalarvectorproduct, 183
 - mqc_vectorcomplexproduct, 183
 - mqc_vectorintegerproduct, 184
 - mqc_vectorrealproduct, 184
 - mqc_vectorscalarproduct, 184
- mqc_algebra::operator(+), 189
 - mqc_complexscalaradd, 189
 - mqc_integerscalaradd, 190
 - mqc_matrixmatrixsum, 190
 - mqc_realscalaradd, 190
 - mqc_scalaradd, 191
 - mqc_scalarcomplexadd, 192
 - mqc_scalarintegeradd, 193
 - mqc_scalarrealadd, 193
 - mqc_scalarvectorsum, 194
 - mqc_vectorvectorsum, 194
- mqc_algebra::operator(-), 195
 - mqc_complexscalarsubtract, 195
 - mqc_integerscalarsubtract, 196
 - mqc_matrixmatrixsubtract, 197
 - mqc_realscalarsubtract, 197
 - mqc_scalarcomplexsubtract, 197
 - mqc_scalarintegersubtract, 198
 - mqc_scalarrealsubtract, 199
 - mqc_scalarsubtract, 200
 - mqc_scalarvectordifference, 200
 - mqc_vectorvectordifference, 200
- mqc_algebra::operator(.dot.), 201
 - mqc_matrixmatrixdotproduct, 201
 - mqc_matrixvectordotproduct, 201
 - mqc_vectormatrixdotproduct, 201
 - mqc_vectorvectordotproduct, 201
- mqc_algebra::operator(.eq.), 202
 - mqc_scalareq, 202
- mqc_algebra::operator(.ewd.), 203
 - mqc_elementmatrixdivide, 203
- mqc_algebra::operator(.ewp.), 203
 - mqc_elementmatrixproduct, 203
 - mqc_elementvectorproduct, 204
- mqc_algebra::operator(.ge.), 204
 - mqc_scalarge, 204
- mqc_algebra::operator(.gt.), 205
 - mqc_integergtscalar, 205
 - mqc_realgtscalar, 206
 - mqc_scalargt, 207
 - mqc_scalargtinteger, 208
 - mqc_scalargtreal, 209
- mqc_algebra::operator(.le.), 210
 - mqc_integerlescalar, 210
 - mqc_reallescalar, 211
 - mqc_scalarle, 211
 - mqc_scalarleinteger, 212
 - mqc_scalarlereal, 213
- mqc_algebra::operator(.lt.), 214
 - mqc_realltscalar, 214
 - mqc_scalarlt, 215
 - mqc_scalarltreal, 216
- mqc_algebra::operator(.ne.), 217
 - mqc_scalarne, 217
- mqc_algebra::operator(.outer.), 218
 - mqc_outer, 218
- mqc_algebra::operator(.x.), 218
 - mqc_crossproduct, 218
- mqc_algebra::operator(/), 219
 - mqc_complexscalardivide, 219
 - mqc_integerscalardivide, 220
 - mqc_realscalardivide, 220
 - mqc_scalarcomplexdivide, 221
 - mqc_scalardivide, 222
 - mqc_scalarintegerdivide, 222
 - mqc_scalarrealdivide, 223
 - mqc_vectorcomplexdivide, 224
 - mqc_vectorintegerdivide, 224
 - mqc_vectorrealdivide, 224
 - mqc_vectorscalardivide, 224
- mqc_algebra::real, 225
 - mqc_scalar_complex_realpart, 225
 - mqc_vector_complex_realpart, 225
- mqc_algebra::sin, 226
 - mqc_scalar_sin, 226
- mqc_algebra::sqrt, 226
 - mqc_scalar_sqrt, 227
- mqc_algebra::tan, 227

- mqc_scalar_tan, 227
- mqc_algebra::transpose, 229
 - mqc_matrix_transpose, 229
 - mqc_vector_transpose, 229
- mqc_allocate_matrix
 - mqc_algebra, 19
- mqc_allocate_r4tensor
 - mqc_algebra, 19
- mqc_allocate_scalar
 - mqc_algebra, 19
- mqc_allocate_vector
 - mqc_algebra, 20
- mqc_build_ci_hamiltonian
 - mqc_est, 104
- mqc_complexscalaradd
 - mqc_algebra, 20
 - mqc_algebra::operator(+), 189
- mqc_complexscalardivide
 - mqc_algebra, 21
 - mqc_algebra::operator(/), 219
- mqc_complexscalarmultiply
 - mqc_algebra, 22
 - mqc_algebra::operator(*), 178
- mqc_complexscalarsubtract
 - mqc_algebra, 23
 - mqc_algebra::operator(-), 195
- mqc_complexvectorproduct
 - mqc_algebra, 23
 - mqc_algebra::operator(*), 178
- mqc_crossproduct
 - mqc_algebra, 23
 - mqc_algebra::operator(.x.), 218
- mqc_deallocate_matrix
 - mqc_algebra, 24
- mqc_deallocate_r4tensor
 - mqc_algebra, 24
- mqc_deallocate_scalar
 - mqc_algebra, 24
- mqc_deallocate_vector
 - mqc_algebra, 25
- mqc_eigenvalue_eigenvalue_dotproduct
 - mqc_est, 104
 - mqc_est::dot_product, 139
- mqc_eigenvalues_add_name
 - mqc_est, 105
- mqc_eigenvalues_allocate
 - mqc_est, 105
- mqc_eigenvalues_array_name
 - mqc_est, 105
- mqc_eigenvalues_array_type
 - mqc_est, 105
- mqc_eigenvalues_at
 - mqc_est, 105
- mqc_eigenvalues_dimension
 - mqc_est, 105
- mqc_eigenvalues_eigenvalues_multiply
 - mqc_est, 106
- mqc_eigenvalues_has_alpha
 - mqc_est, 106
- mqc_eigenvalues_has_beta
 - mqc_est, 106
- mqc_eigenvalues_integral_multiply
 - mqc_est, 106
 - mqc_est::matmul, 141
- mqc_eigenvalues_isallocated
 - mqc_est, 106
- mqc_eigenvalues_output_array
 - mqc_est, 106
 - mqc_est::assignment(=), 131
- mqc_eigenvalues_output_block
 - mqc_est, 107
- mqc_elementmatrixdivide
 - mqc_algebra, 25
 - mqc_algebra::operator(.ewd.), 203
- mqc_elementmatrixproduct
 - mqc_algebra, 25
 - mqc_algebra::operator(.ewp.), 203
- mqc_elementvectorproduct
 - mqc_algebra, 25
 - mqc_algebra::operator(.ewp.), 204
- mqc_eri_integral_contraction
 - mqc_est, 107
 - mqc_est::contraction, 136
- mqc_est, 102
 - gen_det_str, 104
 - get_one_gamma_matrix, 104
 - mqc_build_ci_hamiltonian, 104
 - mqc_eigenvalue_eigenvalue_dotproduct, 104
 - mqc_eigenvalues_add_name, 105
 - mqc_eigenvalues_allocate, 105
 - mqc_eigenvalues_array_name, 105
 - mqc_eigenvalues_array_type, 105
 - mqc_eigenvalues_at, 105
 - mqc_eigenvalues_dimension, 105
 - mqc_eigenvalues_eigenvalues_multiply, 106
 - mqc_eigenvalues_has_alpha, 106
 - mqc_eigenvalues_has_beta, 106
 - mqc_eigenvalues_integral_multiply, 106
 - mqc_eigenvalues_isallocated, 106
 - mqc_eigenvalues_output_array, 106
 - mqc_eigenvalues_output_block, 107
 - mqc_eri_integral_contraction, 107
 - mqc_integral_add_name, 107
 - mqc_integral_allocate, 107
 - mqc_integral_array_name, 107
 - mqc_integral_array_type, 108
 - mqc_integral_at, 108

- mqc_integral_conjugate_transpose, 108
- mqc_integral_delete_energy_list, 108
- mqc_integral_difference, 108
- mqc_integral_dimension, 108
- mqc_integral_eigenvalues_multiply, 109
- mqc_integral_get_energy_list, 109
- mqc_integral_has_alpha, 109
- mqc_integral_has_alphabeta, 109
- mqc_integral_has_beta, 109
- mqc_integral_has_betaalpha, 109
- mqc_integral_identity, 110
- mqc_integral_initialize, 110
- mqc_integral_integral_multiply, 110
- mqc_integral_isallocated, 110
- mqc_integral_matrix_multiply, 110
- mqc_integral_norm, 111
- mqc_integral_output_array, 111
- mqc_integral_output_block, 111
- mqc_integral_output_orbitals, 111
- mqc_integral_scalar_multiply, 111
- mqc_integral_set_energy_list, 112
- mqc_integral_sum, 112
- mqc_integral_swap_orbitals, 112
- mqc_integral_transpose, 112
- mqc_matrix_integral_multiply, 112
- mqc_matrix_spinblockghf, 113
- mqc_matrix_undospinblockghf_eigenvalues, 113
- mqc_matrix_undospinblockghf_integral, 113
- mqc_print_eigenvalues, 113
- mqc_print_integral, 113
- mqc_print_twoeris, 114
- mqc_print_wavefunction, 114
- mqc_scalar_integral_multiply, 114
- mqc_scf_eigenvalues_power, 114
- mqc_scf_integral_contraction, 114
- mqc_scf_integral_determinant, 115
- mqc_scf_integral_diagonalize, 115
- mqc_scf_integral_generalized_eigensystem, 115
- mqc_scf_integral_inverse, 115
- mqc_scf_integral_trace, 115
- mqc_scf_transformation_matrix, 115
- mqc_twoeris_allocate, 116
- mqc_twoeris_at, 116
- slater_condon, 116
- twoeri_trans, 116
- mqc_est::assignment(=), 131
 - mqc_eigenvalues_output_array, 131
 - mqc_integral_output_array, 131
- mqc_est::contraction, 136
 - mqc_eri_integral_contraction, 136
 - mqc_scf_integral_contraction, 136
- mqc_est::dagger, 138
 - mqc_integral_conjugate_transpose, 138
- mqc_est::dot_product, 139
 - mqc_eigenvalue_eigenvalue_dotproduct, 139
- mqc_est::matmul, 141
 - mqc_eigenvalues_eigenvalues_multiply, 141
 - mqc_eigenvalues_integral_multiply, 141
 - mqc_integral_eigenvalues_multiply, 141
 - mqc_integral_integral_multiply, 141
 - mqc_integral_matrix_multiply, 142
 - mqc_matrix_integral_multiply, 142
- mqc_est::mqc_determinant, 144
 - nalpstr, 145
 - nbetstr, 145
 - ndets, 145
 - order, 145
 - strings, 145
- mqc_est::mqc_determinant_string, 145
 - alpha, 146
 - beta, 146
- mqc_est::mqc_matrix_undospinblockghf, 155
 - mqc_matrix_undospinblockghf_eigenvalues, 155
 - mqc_matrix_undospinblockghf_integral, 155
- mqc_est::mqc_print, 157
 - mqc_print_eigenvalues, 158
 - mqc_print_integral, 158
 - mqc_print_twoeris, 158
 - mqc_print_wavefunction, 158
- mqc_est::mqc_pscf_wavefunction, 159
 - nactive, 159
 - ncore, 159
 - nfrz, 159
 - nval, 159
 - pscf_amplitudes, 160
 - pscf_energies, 160
- mqc_est::mqc_scf_eigenvalues, 163
 - addlabel, 163
 - at, 163
 - getblock, 163
 - getlabel, 163
 - power, 163
 - print, 163
- mqc_est::mqc_scf_integral, 164
 - addlabel, 164
 - deleteelist, 164
 - det, 164
 - diag, 165
 - eigensys, 165
 - getblock, 165
 - getelist, 165
 - getlabel, 165
 - identity, 165
 - init, 165
 - inv, 166
 - norm, 166
 - orbitals, 166
 - print, 166

- setelist, 166
 - swap, 166
 - trace, 166
- mqc_est::mqc_twoeris, 168
 - print, 168
- mqc_est::mqc_wavefunction, 173
 - basis, 174
 - charge, 174
 - core_hamiltonian, 174
 - density_matrix, 175
 - fock_matrix, 175
 - mo_coefficients, 175
 - mo_energies, 175
 - mo_symmetries, 175
 - multiplicity, 175
 - nalpha, 175
 - nbasis, 176
 - nbeta, 176
 - nelectrons, 176
 - overlap_matrix, 176
 - print, 174
 - scf_density_matrix, 176
 - symmetry, 176
 - wf_complex, 176
 - wf_type, 177
- mqc_est::operator(*), 184
 - mqc_integral_scalar_multiply, 184
 - mqc_scalar_integral_multiply, 185
- mqc_est::operator(+), 188
 - mqc_integral_sum, 188
- mqc_est::operator(-), 194
 - mqc_integral_difference, 195
- mqc_est::transpose, 228
 - mqc_integral_transpose, 228
- mqc_givens_matrix
 - mqc_algebra, 25
- mqc_input_complex_scalar
 - mqc_algebra, 25
 - mqc_algebra::assignment(=), 124
- mqc_input_integer_scalar
 - mqc_algebra, 26
 - mqc_algebra::assignment(=), 124
- mqc_input_real_scalar
 - mqc_algebra, 27
 - mqc_algebra::assignment(=), 125
- mqc_integertscalar
 - mqc_algebra, 28
 - mqc_algebra::operator(.gt.), 205
- mqc_integerlescalar
 - mqc_algebra, 28
 - mqc_algebra::operator(.le.), 210
- mqc_integerscalaradd
 - mqc_algebra, 29
 - mqc_algebra::operator(+), 190
- mqc_integerscalardivide
 - mqc_algebra, 30
 - mqc_algebra::operator(/), 220
- mqc_integerscalarmultiply
 - mqc_algebra, 30
 - mqc_algebra::operator(*), 178
- mqc_integerscalarsubtract
 - mqc_algebra, 31
 - mqc_algebra::operator(-), 196
- mqc_integervectorproduct
 - mqc_algebra, 32
 - mqc_algebra::operator(*), 179
- mqc_integral_add_name
 - mqc_est, 107
- mqc_integral_allocate
 - mqc_est, 107
- mqc_integral_array_name
 - mqc_est, 107
- mqc_integral_array_type
 - mqc_est, 108
- mqc_integral_at
 - mqc_est, 108
- mqc_integral_conjugate_transpose
 - mqc_est, 108
 - mqc_est::dagger, 138
- mqc_integral_delete_energy_list
 - mqc_est, 108
- mqc_integral_difference
 - mqc_est, 108
 - mqc_est::operator(-), 195
- mqc_integral_dimension
 - mqc_est, 108
- mqc_integral_eigenvalues_multiply
 - mqc_est, 109
 - mqc_est::matmul, 141
- mqc_integral_get_energy_list
 - mqc_est, 109
- mqc_integral_has_alpha
 - mqc_est, 109
- mqc_integral_has_alphabeta
 - mqc_est, 109
- mqc_integral_has_beta
 - mqc_est, 109
- mqc_integral_has_betaalpha
 - mqc_est, 109
- mqc_integral_identity
 - mqc_est, 110
- mqc_integral_initialize
 - mqc_est, 110
- mqc_integral_integral_multiply
 - mqc_est, 110
 - mqc_est::matmul, 141
- mqc_integral_isallocated
 - mqc_est, 110

mqc_integral_matrix_multiply
 mqc_est, 110
 mqc_est::matmul, 142
 mqc_integral_norm
 mqc_est, 111
 mqc_integral_output_array
 mqc_est, 111
 mqc_est::assignment(=), 131
 mqc_integral_output_block
 mqc_est, 111
 mqc_integral_output_orbitals
 mqc_est, 111
 mqc_integral_scalar_multiply
 mqc_est, 111
 mqc_est::operator(*), 184
 mqc_integral_set_energy_list
 mqc_est, 112
 mqc_integral_sum
 mqc_est, 112
 mqc_est::operator(+), 188
 mqc_integral_swap_orbitals
 mqc_est, 112
 mqc_integral_transpose
 mqc_est, 112
 mqc_est::transpose, 228
 mqc_length_vector
 mqc_algebra, 32
 mqc_matrix_cast_complex
 mqc_algebra, 32
 mqc_algebra::mqc_cast_complex, 143
 mqc_matrix_cast_real
 mqc_algebra, 32
 mqc_algebra::mqc_cast_real, 144
 mqc_matrix_columns
 mqc_algebra, 32
 mqc_matrix_conjugate_transpose
 mqc_algebra, 32
 mqc_algebra::dagger, 138
 mqc_matrix_copy_complex2int
 mqc_algebra, 33
 mqc_matrix_copy_complex2real
 mqc_algebra, 33
 mqc_matrix_copy_int2complex
 mqc_algebra, 33
 mqc_matrix_copy_int2real
 mqc_algebra, 33
 mqc_matrix_copy_real2complex
 mqc_algebra, 33
 mqc_matrix_copy_real2int
 mqc_algebra, 33
 mqc_matrix_determinant
 mqc_algebra, 34
 mqc_matrix_diag2full
 mqc_algebra, 34
 mqc_matrix_diag2symm
 mqc_algebra, 34
 mqc_matrix_diagmatrix_put_complex
 mqc_algebra, 34
 mqc_algebra::mqc_matrix_diagmatrix_put, 153
 mqc_matrix_diagmatrix_put_integer
 mqc_algebra, 34
 mqc_algebra::mqc_matrix_diagmatrix_put, 153
 mqc_matrix_diagmatrix_put_real
 mqc_algebra, 34
 mqc_algebra::mqc_matrix_diagmatrix_put, 153
 mqc_matrix_diagmatrix_put_vector
 mqc_algebra, 35
 mqc_algebra::mqc_matrix_diagmatrix_put, 153
 mqc_matrix_diagonalize
 mqc_algebra, 35
 mqc_matrix_full2diag
 mqc_algebra, 35
 mqc_matrix_full2symm
 mqc_algebra, 35
 mqc_matrix_generalized_eigensystem
 mqc_algebra, 35
 mqc_matrix_havecomplex
 mqc_algebra, 35
 mqc_algebra::mqc_have_complex, 146
 mqc_matrix_havediagonal
 mqc_algebra, 36
 mqc_matrix_havefull
 mqc_algebra, 36
 mqc_matrix_haveinteger
 mqc_algebra, 36
 mqc_algebra::mqc_have_int, 147
 mqc_matrix_havereal
 mqc_algebra, 36
 mqc_algebra::mqc_have_real, 147
 mqc_matrix_havesymmetric
 mqc_algebra, 36
 mqc_matrix_identity
 mqc_algebra, 36
 mqc_matrix_initialize
 mqc_algebra, 37
 mqc_matrix_integral_multiply
 mqc_est, 112
 mqc_est::matmul, 142
 mqc_matrix_inverse
 mqc_algebra, 37
 mqc_matrix_isallocated
 mqc_algebra, 37
 mqc_matrix_matrix_at
 mqc_algebra, 37
 mqc_matrix_matrix_contraction
 mqc_algebra, 38
 mqc_algebra::contraction, 136
 mqc_matrix_matrix_put

- mqc_algebra, 38
- mqc_matrix_norm
 - mqc_algebra, 38
- mqc_matrix_rms_max
 - mqc_algebra, 39
- mqc_matrix_rows
 - mqc_algebra, 39
- mqc_matrix_scalar_at
 - mqc_algebra, 39
- mqc_matrix_scalar_put
 - mqc_algebra, 39
- mqc_matrix_set
 - mqc_algebra, 39
- mqc_matrix_spinblockghf
 - mqc_est, 113
- mqc_matrix_sqrt
 - mqc_algebra, 40
- mqc_matrix_storagetype
 - mqc_algebra, 40
- mqc_matrix_svd
 - mqc_algebra, 40
- mqc_matrix_symm2diag
 - mqc_algebra, 40
- mqc_matrix_symm2full
 - mqc_algebra, 40
- mqc_matrix_symm2full_func
 - mqc_algebra, 41
- mqc_matrix_symmetrize
 - mqc_algebra, 41
- mqc_matrix_symmmatrix_put_complex
 - mqc_algebra, 41
 - mqc_algebra::mqc_matrix_symmmatrix_put, 154
- mqc_matrix_symmmatrix_put_integer
 - mqc_algebra, 41
 - mqc_algebra::mqc_matrix_symmmatrix_put, 154
- mqc_matrix_symmmatrix_put_real
 - mqc_algebra, 41
 - mqc_algebra::mqc_matrix_symmmatrix_put, 154
- mqc_matrix_symmsymmr4tensor_put_complex
 - mqc_algebra, 41
- mqc_matrix_symmsymmr4tensor_put_real
 - mqc_algebra, 42
- mqc_matrix_test_diagonal
 - mqc_algebra, 42
- mqc_matrix_test_symmetric
 - mqc_algebra, 42
- mqc_matrix_trace
 - mqc_algebra, 42
- mqc_matrix_transpose
 - mqc_algebra, 42
 - mqc_algebra::transpose, 229
- mqc_matrix_undospinblockghf_eigenvalues
 - mqc_est, 113
 - mqc_est::mqc_matrix_undospinblockghf, 155
- mqc_matrix_undospinblockghf_integral
 - mqc_est, 113
 - mqc_est::mqc_matrix_undospinblockghf, 155
- mqc_matrix_vector_at
 - mqc_algebra, 42
- mqc_matrix_vector_put
 - mqc_algebra, 43
- mqc_matrixmatrixdotproduct
 - mqc_algebra, 43
 - mqc_algebra::matmul, 140
 - mqc_algebra::operator(.dot.), 201
- mqc_matrixmatrixproduct
 - mqc_algebra, 43
 - mqc_algebra::operator(*), 179
- mqc_matrixmatrixsubtract
 - mqc_algebra, 43
 - mqc_algebra::operator(-), 197
- mqc_matrixmatrixsum
 - mqc_algebra, 43
 - mqc_algebra::operator(+), 190
- mqc_matrixscalarproduct
 - mqc_algebra, 44
 - mqc_algebra::operator(*), 179
- mqc_matrixvectordotproduct
 - mqc_algebra, 44
 - mqc_algebra::matmul, 140
 - mqc_algebra::operator(.dot.), 201
- mqc_outer
 - mqc_algebra, 44
 - mqc_algebra::operator(.outer.), 218
- mqc_output_complex_scalar
 - mqc_algebra, 44
 - mqc_algebra::assignment(=), 126
- mqc_output_integer_scalar
 - mqc_algebra, 45
 - mqc_algebra::assignment(=), 126
- mqc_output_mqcscalar_scalar
 - mqc_algebra, 46
 - mqc_algebra::assignment(=), 127
- mqc_output_real_scalar
 - mqc_algebra, 46
 - mqc_algebra::assignment(=), 128
- mqc_print_eigenvalues
 - mqc_est, 113
 - mqc_est::mqc_print, 158
- mqc_print_integral
 - mqc_est, 113
 - mqc_est::mqc_print, 158
- mqc_print_matrix_algebra1
 - mqc_algebra, 47
 - mqc_algebra::mqc_print, 155
- mqc_print_r4tensor_algebra1
 - mqc_algebra, 47
 - mqc_algebra::mqc_print, 156

mqc_print_scalar_algebra1
 mqc_algebra, 47
 mqc_algebra::mqc_print, 156
 mqc_print_twoeris
 mqc_est, 114
 mqc_est::mqc_print, 158
 mqc_print_vector_algebra1
 mqc_algebra, 48
 mqc_algebra::mqc_print, 157
 mqc_print_wavefunction
 mqc_est, 114
 mqc_est::mqc_print, 158
 mqc_r4tensor_at
 mqc_algebra, 49
 mqc_r4tensor_havecomplex
 mqc_algebra, 49
 mqc_r4tensor_haveinteger
 mqc_algebra, 49
 mqc_r4tensor_havereal
 mqc_algebra, 49
 mqc_r4tensor_initialize
 mqc_algebra, 49
 mqc_r4tensor_put
 mqc_algebra, 50
 mqc_realgtscalar
 mqc_algebra, 50
 mqc_algebra::operator(.gt.), 206
 mqc_reallscalar
 mqc_algebra, 51
 mqc_algebra::operator(.le.), 211
 mqc_realltscalar
 mqc_algebra, 51
 mqc_algebra::operator(.lt.), 214
 mqc_realscalaradd
 mqc_algebra, 52
 mqc_algebra::operator(+), 190
 mqc_realscalardivide
 mqc_algebra, 53
 mqc_algebra::operator(/), 220
 mqc_realscalarmultiply
 mqc_algebra, 54
 mqc_algebra::operator(*), 180
 mqc_realscalarsubtract
 mqc_algebra, 54
 mqc_algebra::operator(-), 197
 mqc_realvectorproduct
 mqc_algebra, 55
 mqc_algebra::operator(*), 180
 mqc_scalar_acos
 mqc_algebra, 55
 mqc_algebra::acos, 120
 mqc_scalar_asin
 mqc_algebra, 56
 mqc_algebra::asin, 122
 mqc_scalar_atan
 mqc_algebra, 56
 mqc_algebra::atan, 132
 mqc_scalar_atan2
 mqc_algebra, 57
 mqc_algebra::atan2, 133
 mqc_scalar_cmplx
 mqc_algebra, 57
 mqc_algebra::cmplx, 134
 mqc_scalar_complex_conjugate
 mqc_algebra, 58
 mqc_algebra::conjg, 135
 mqc_scalar_complex_imagpart
 mqc_algebra, 59
 mqc_algebra::aimag, 121
 mqc_scalar_complex_realpart
 mqc_algebra, 59
 mqc_algebra::real, 225
 mqc_scalar_cos
 mqc_algebra, 60
 mqc_algebra::cos, 137
 mqc_scalar_get_abs_value
 mqc_algebra, 61
 mqc_algebra::abs, 119
 mqc_scalar_get_intrinsic_complex
 mqc_algebra, 61
 mqc_scalar_get_intrinsic_integer
 mqc_algebra, 62
 mqc_scalar_get_intrinsic_real
 mqc_algebra, 62
 mqc_scalar_get_random_value
 mqc_algebra, 63
 mqc_scalar_havecomplex
 mqc_algebra, 64
 mqc_scalar_haveinteger
 mqc_algebra, 64
 mqc_scalar_havereal
 mqc_algebra, 65
 mqc_scalar_integral_multiply
 mqc_est, 114
 mqc_est::operator(*), 185
 mqc_scalar_isallocated
 mqc_algebra, 66
 mqc_scalar_sin
 mqc_algebra, 66
 mqc_algebra::sin, 226
 mqc_scalar_sqrt
 mqc_algebra, 67
 mqc_algebra::sqrt, 227
 mqc_scalar_tan
 mqc_algebra, 67
 mqc_algebra::tan, 227
 mqc_scalaradd
 mqc_algebra, 68

- mqc_algebra::operator(+), 191
- mqc_scalarcomplexadd
 - mqc_algebra, 69
 - mqc_algebra::operator(+), 192
- mqc_scalarcomplexdivide
 - mqc_algebra, 69
 - mqc_algebra::operator(/), 221
- mqc_scalarcomplexexponent
 - mqc_algebra, 70
 - mqc_algebra::operator(**), 185
- mqc_scalarcomplexmultiply
 - mqc_algebra, 71
 - mqc_algebra::operator(*), 180
- mqc_scalarcomplexsubtract
 - mqc_algebra, 71
 - mqc_algebra::operator(-), 197
- mqc_scalardivide
 - mqc_algebra, 72
 - mqc_algebra::operator(/), 222
- mqc_scalareq
 - mqc_algebra, 73
 - mqc_algebra::operator(.eq.), 202
- mqc_scalarexponent
 - mqc_algebra, 73
 - mqc_algebra::operator(**), 186
- mqc_scalarge
 - mqc_algebra, 74
 - mqc_algebra::operator(.ge.), 204
- mqc_scalargt
 - mqc_algebra, 75
 - mqc_algebra::operator(.gt.), 207
- mqc_scalargtinteger
 - mqc_algebra, 75
 - mqc_algebra::operator(.gt.), 208
- mqc_scalargtreal
 - mqc_algebra, 76
 - mqc_algebra::operator(.gt.), 209
- mqc_scalarintegeradd
 - mqc_algebra, 77
 - mqc_algebra::operator(+), 193
- mqc_scalarintegerdivide
 - mqc_algebra, 78
 - mqc_algebra::operator(/), 222
- mqc_scalarintegerexponent
 - mqc_algebra, 78
 - mqc_algebra::operator(**), 187
- mqc_scalarintegermultiply
 - mqc_algebra, 79
 - mqc_algebra::operator(*), 181
- mqc_scalarintegersubtract
 - mqc_algebra, 80
 - mqc_algebra::operator(-), 198
- mqc_scalarle
 - mqc_algebra, 80
 - mqc_algebra::operator(.le.), 211
- mqc_scalarleinteger
 - mqc_algebra, 81
 - mqc_algebra::operator(.le.), 212
- mqc_scalarlereal
 - mqc_algebra, 82
 - mqc_algebra::operator(.le.), 213
- mqc_scalarlt
 - mqc_algebra, 83
 - mqc_algebra::operator(.lt.), 215
- mqc_scalarltreal
 - mqc_algebra, 83
 - mqc_algebra::operator(.lt.), 216
- mqc_scalarmatrixproduct
 - mqc_algebra, 84
 - mqc_algebra::operator(*), 182
- mqc_scalarmultiply
 - mqc_algebra, 84
 - mqc_algebra::operator(*), 182
- mqc_scalarne
 - mqc_algebra, 85
 - mqc_algebra::operator(.ne.), 217
- mqc_scalarrealadd
 - mqc_algebra, 86
 - mqc_algebra::operator(+), 193
- mqc_scalarrealddivide
 - mqc_algebra, 87
 - mqc_algebra::operator(/), 223
- mqc_scalarrealexponent
 - mqc_algebra, 87
 - mqc_algebra::operator(**), 187
- mqc_scalarrealmultiply
 - mqc_algebra, 88
 - mqc_algebra::operator(*), 183
- mqc_scalarrealsubtract
 - mqc_algebra, 89
 - mqc_algebra::operator(-), 199
- mqc_scalarsubtract
 - mqc_algebra, 89
 - mqc_algebra::operator(-), 200
- mqc_scalarvectordifference
 - mqc_algebra, 90
 - mqc_algebra::operator(-), 200
- mqc_scalarvectorproduct
 - mqc_algebra, 90
 - mqc_algebra::operator(*), 183
- mqc_scalarvectorsum
 - mqc_algebra, 90
 - mqc_algebra::operator(+), 194
- mqc_scf_eigenvalues_power
 - mqc_est, 114
- mqc_scf_integral_contraction
 - mqc_est, 114
 - mqc_est::contraction, 136

mqc_scf_integral_determinant
 mqc_est, 115
 mqc_scf_integral_diagonalize
 mqc_est, 115
 mqc_scf_integral_generalized_eigensystem
 mqc_est, 115
 mqc_scf_integral_inverse
 mqc_est, 115
 mqc_scf_integral_trace
 mqc_est, 115
 mqc_scf_transformation_matrix
 mqc_est, 115
 mqc_set_array2tensor
 mqc_algebra, 90
 mqc_algebra::assignment(=), 128
 mqc_set_array2vector_complex
 mqc_algebra, 91
 mqc_algebra::assignment(=), 128
 mqc_algebra::mqc_set_array2vector, 167
 mqc_set_array2vector_integer
 mqc_algebra, 91
 mqc_algebra::assignment(=), 129
 mqc_algebra::mqc_set_array2vector, 167
 mqc_set_array2vector_real
 mqc_algebra, 91
 mqc_algebra::assignment(=), 129
 mqc_algebra::mqc_set_array2vector, 167
 mqc_set_complexarray2matrix
 mqc_algebra, 91
 mqc_algebra::assignment(=), 129
 mqc_set_integerarray2matrix
 mqc_algebra, 91
 mqc_algebra::assignment(=), 129
 mqc_set_matrix2complexarray
 mqc_algebra, 91
 mqc_algebra::assignment(=), 129
 mqc_set_matrix2integerarray
 mqc_algebra, 92
 mqc_algebra::assignment(=), 129
 mqc_set_matrix2matrix
 mqc_algebra, 92
 mqc_algebra::assignment(=), 130
 mqc_set_matrix2realarray
 mqc_algebra, 92
 mqc_algebra::assignment(=), 130
 mqc_set_realarray2matrix
 mqc_algebra, 92
 mqc_algebra::assignment(=), 130
 mqc_set_vector2complexarray
 mqc_algebra, 92
 mqc_algebra::assignment(=), 130
 mqc_set_vector2integerarray
 mqc_algebra, 92
 mqc_algebra::assignment(=), 130
 mqc_set_vector2realarray
 mqc_algebra, 93
 mqc_algebra::assignment(=), 130
 mqc_set_vector2vector
 mqc_algebra, 93
 mqc_algebra::assignment(=), 131
 mqc_twoeris_allocate
 mqc_est, 116
 mqc_twoeris_at
 mqc_est, 116
 mqc_vector2diagmatrix
 mqc_algebra, 93
 mqc_vector_abs
 mqc_algebra, 93
 mqc_algebra::abs, 120
 mqc_vector_argsort
 mqc_algebra, 93
 mqc_vector_cast_complex
 mqc_algebra, 93
 mqc_algebra::mqc_cast_complex, 143
 mqc_vector_cast_real
 mqc_algebra, 94
 mqc_algebra::mqc_cast_real, 144
 mqc_vector_cmplx
 mqc_algebra, 94
 mqc_algebra::cmplx, 134
 mqc_vector_complex_imagpart
 mqc_algebra, 94
 mqc_algebra::aimag, 122
 mqc_vector_complex_realpart
 mqc_algebra, 94
 mqc_algebra::real, 225
 mqc_vector_conjugate_transpose
 mqc_algebra, 94
 mqc_algebra::dagger, 138
 mqc_vector_copy_complex2int
 mqc_algebra, 94
 mqc_vector_copy_complex2real
 mqc_algebra, 95
 mqc_vector_copy_int2complex
 mqc_algebra, 95
 mqc_vector_copy_int2real
 mqc_algebra, 95
 mqc_vector_copy_real2complex
 mqc_algebra, 95
 mqc_vector_copy_real2int
 mqc_algebra, 95
 mqc_vector_havecomplex
 mqc_algebra, 95
 mqc_algebra::mqc_have_complex, 146
 mqc_vector_haveinteger
 mqc_algebra, 96
 mqc_algebra::mqc_have_int, 147
 mqc_vector_havereal

- mqc_algebra, 96
- mqc_algebra::mqc_have_real, 148
- mqc_vector_initialize
 - mqc_algebra, 96
- mqc_vector_isallocated
 - mqc_algebra, 96
- mqc_vector_iscolumn
 - mqc_algebra, 96
- mqc_vector_maxloc
 - mqc_algebra, 96
- mqc_vector_maxval
 - mqc_algebra, 97
- mqc_vector_minloc
 - mqc_algebra, 97
- mqc_vector_minval
 - mqc_algebra, 97
- mqc_vector_norm
 - mqc_algebra, 97
- mqc_vector_pop
 - mqc_algebra, 97
- mqc_vector_power
 - mqc_algebra, 97
- mqc_vector_push
 - mqc_algebra, 98
- mqc_vector_scalar_at
 - mqc_algebra, 98
- mqc_vector_scalar_increment
 - mqc_algebra, 98
- mqc_vector_scalar_put
 - mqc_algebra, 98
- mqc_vector_shift
 - mqc_algebra, 98
- mqc_vector_sort
 - mqc_algebra, 98
- mqc_vector_sqrt
 - mqc_algebra, 99
- mqc_vector_transpose
 - mqc_algebra, 99
 - mqc_algebra::transpose, 229
- mqc_vector_unshift
 - mqc_algebra, 99
- mqc_vector_vector_at
 - mqc_algebra, 99
- mqc_vector_vector_put
 - mqc_algebra, 99
- mqc_vectorcomplexdivide
 - mqc_algebra, 99
 - mqc_algebra::operator(/), 224
- mqc_vectorcomplexproduct
 - mqc_algebra, 100
 - mqc_algebra::operator(*), 183
- mqc_vectorintegerdivide
 - mqc_algebra, 100
 - mqc_algebra::operator(/), 224
- mqc_vectorintegerproduct
 - mqc_algebra, 100
 - mqc_algebra::operator(*), 184
- mqc_vectormatrixdotproduct
 - mqc_algebra, 100
 - mqc_algebra::matmul, 140
 - mqc_algebra::operator(.dot.), 201
- mqc_vectorrealdivide
 - mqc_algebra, 100
 - mqc_algebra::operator(/), 224
- mqc_vectorrealproduct
 - mqc_algebra, 100
 - mqc_algebra::operator(*), 184
- mqc_vectorscalardivide
 - mqc_algebra, 101
 - mqc_algebra::operator(/), 224
- mqc_vectorscalarproduct
 - mqc_algebra, 101
 - mqc_algebra::operator(*), 184
- mqc_vectorvectordifference
 - mqc_algebra, 101
 - mqc_algebra::operator(-), 200
- mqc_vectorvectordotproduct
 - mqc_algebra, 101
 - mqc_algebra::dot_product, 139
 - mqc_algebra::operator(.dot.), 201
- mqc_vectorvectorsum
 - mqc_algebra, 101
 - mqc_algebra::operator(+), 194
- multiplicity
 - mqc_est::mqc_wavefunction, 175
- nactive
 - mqc_est::mqc_pscf_wavefunction, 159
- nalpha
 - mqc_est::mqc_wavefunction, 175
- nalpstr
 - mqc_est::mqc_determinant, 145
- nbasis
 - mqc_est::mqc_wavefunction, 176
- nbeta
 - mqc_est::mqc_wavefunction, 176
- nbetstr
 - mqc_est::mqc_determinant, 145
- ncore
 - mqc_est::mqc_pscf_wavefunction, 159
- ndets
 - mqc_est::mqc_determinant, 145
- nelectrons
 - mqc_est::mqc_wavefunction, 176
- nfrz
 - mqc_est::mqc_pscf_wavefunction, 159
- norm
 - mqc_algebra::mqc_matrix, 150

- mqc_algebra::mqc_vector, 170
 - mqc_est::mqc_scf_integral, 166
- nval
 - mqc_est::mqc_pscf_wavefunction, 159
- orbitals
 - mqc_est::mqc_scf_integral, 166
- order
 - mqc_est::mqc_determinant, 145
- overlap_matrix
 - mqc_est::mqc_wavefunction, 176
- pop
 - mqc_algebra::mqc_vector, 170
- power
 - mqc_algebra::mqc_vector, 171
 - mqc_est::mqc_scf_eigenvalues, 163
- print
 - mqc_algebra::mqc_matrix, 150
 - mqc_algebra::mqc_r4tensor, 161
 - mqc_algebra::mqc_scalar, 162
 - mqc_algebra::mqc_vector, 171
 - mqc_est::mqc_scf_eigenvalues, 163
 - mqc_est::mqc_scf_integral, 166
 - mqc_est::mqc_twoeris, 168
 - mqc_est::mqc_wavefunction, 174
- pscf_amplitudes
 - mqc_est::mqc_pscf_wavefunction, 160
- pscf_energies
 - mqc_est::mqc_pscf_wavefunction, 160
- push
 - mqc_algebra::mqc_vector, 171
- put
 - mqc_algebra::mqc_matrix, 151
 - mqc_algebra::mqc_r4tensor, 161
 - mqc_algebra::mqc_vector, 171
- random
 - mqc_algebra::mqc_scalar, 162
- rmsmax
 - mqc_algebra::mqc_matrix, 151
- rval
 - mqc_algebra::mqc_scalar, 162
- s_type
 - mqc_algebra::mqc_matrix, 151
- scf_density_matrix
 - mqc_est::mqc_wavefunction, 176
- set
 - mqc_algebra::mqc_matrix, 151
- setelist
 - mqc_est::mqc_scf_integral, 166
- shift
 - mqc_algebra::mqc_vector, 171
- size
 - mqc_algebra::mqc_vector, 171
- slater_condon
 - mqc_est, 116
- sort
 - mqc_algebra::mqc_vector, 171
- sqrt
 - mqc_algebra::mqc_matrix, 151
 - mqc_algebra::mqc_vector, 172
- src/mqc_algebra.F03, 231
- src/mqc_est.F03, 239
- strings
 - mqc_est::mqc_determinant, 145
- svd
 - mqc_algebra::mqc_matrix, 151
- swap
 - mqc_est::mqc_scf_integral, 166
- symindexhash
 - mqc_algebra, 101
- symmetry
 - mqc_est::mqc_wavefunction, 176
- trace
 - mqc_algebra::mqc_matrix, 151
 - mqc_est::mqc_scf_integral, 166
- transpose
 - mqc_algebra::mqc_matrix, 152
 - mqc_algebra::mqc_vector, 172
- twoeri_trans
 - mqc_est, 116
- unshift
 - mqc_algebra::mqc_vector, 172
- vat
 - mqc_algebra::mqc_matrix, 152
 - mqc_algebra::mqc_vector, 172
- vecc
 - mqc_algebra::mqc_vector, 173
- veci
 - mqc_algebra::mqc_vector, 173
- vecr
 - mqc_algebra::mqc_vector, 173
- vput
 - mqc_algebra::mqc_matrix, 152
 - mqc_algebra::mqc_vector, 172
- wf_complex
 - mqc_est::mqc_wavefunction, 176
- wf_type
 - mqc_est::mqc_wavefunction, 177