

MQCPack

Generated by Doxygen 1.8.16



<b>1 Modules Index</b>	<b>1</b>
1.1 Modules List	1
<b>2 Data Type Index</b>	<b>3</b>
2.1 Class Hierarchy	3
<b>3 Data Type Index</b>	<b>5</b>
3.1 Data Types List	5
<b>4 File Index</b>	<b>9</b>
4.1 File List	9
<b>5 Module Documentation</b>	<b>11</b>
5.1 mqc_algebra Module Reference	11
5.1.1 Detailed Description	24
5.1.2 Function/Subroutine Documentation	24
5.1.2.1 bin_coeff()	24
5.1.2.2 factorial()	25
5.1.2.3 matrix_symm2sq_complex()	26
5.1.2.4 matrix_symm2sq_integer()	26
5.1.2.5 matrix_symm2sq_real()	27
5.1.2.6 mqc_allocate_matrix()	28
5.1.2.7 mqc_allocate_r4tensor()	29
5.1.2.8 mqc_allocate_scalar()	30
5.1.2.9 mqc_allocate_vector()	30
5.1.2.10 mqc_complexscalaradd()	31
5.1.2.11 mqc_complexscalardivide()	32
5.1.2.12 mqc_complexscalarmultiply()	33
5.1.2.13 mqc_complexscalarsubtract()	33
5.1.2.14 mqc_complexvectorproduct()	34
5.1.2.15 mqc_crossproduct()	35
5.1.2.16 mqc_deallocate_matrix()	36
5.1.2.17 mqc_deallocate_r4tensor()	36
5.1.2.18 mqc_deallocate_scalar()	36
5.1.2.19 mqc_deallocate_vector()	37
5.1.2.20 mqc_elementmatrixdivide()	38
5.1.2.21 mqc_elementmatrixproduct()	38
5.1.2.22 mqc_elementvectorproduct()	39
5.1.2.23 mqc_givens_matrix()	40
5.1.2.24 mqc_input_complex_scalar()	40
5.1.2.25 mqc_input_integer_scalar()	41

5.1.2.26 <code>mqc_input_real_scalar()</code> . . . . .	41
5.1.2.27 <code>mqc_integergtscalar()</code> . . . . .	42
5.1.2.28 <code>mqc_integerlescalar()</code> . . . . .	43
5.1.2.29 <code>mqc_integerscalaradd()</code> . . . . .	44
5.1.2.30 <code>mqc_integerscalardivide()</code> . . . . .	44
5.1.2.31 <code>mqc_integerscalarmultiply()</code> . . . . .	45
5.1.2.32 <code>mqc_integerscalarsubtract()</code> . . . . .	46
5.1.2.33 <code>mqc_integervectorproduct()</code> . . . . .	46
5.1.2.34 <code>mqc_length_vector()</code> . . . . .	47
5.1.2.35 <code>mqc_matrix_cast_complex()</code> . . . . .	48
5.1.2.36 <code>mqc_matrix_cast_integer()</code> . . . . .	48
5.1.2.37 <code>mqc_matrix_cast_real()</code> . . . . .	49
5.1.2.38 <code>mqc_matrix_columns()</code> . . . . .	49
5.1.2.39 <code>mqc_matrix_conjugate_transpose()</code> . . . . .	50
5.1.2.40 <code>mqc_matrix_copy_complex2int()</code> . . . . .	51
5.1.2.41 <code>mqc_matrix_copy_complex2real()</code> . . . . .	51
5.1.2.42 <code>mqc_matrix_copy_int2complex()</code> . . . . .	52
5.1.2.43 <code>mqc_matrix_copy_int2real()</code> . . . . .	52
5.1.2.44 <code>mqc_matrix_copy_real2complex()</code> . . . . .	53
5.1.2.45 <code>mqc_matrix_copy_real2int()</code> . . . . .	54
5.1.2.46 <code>mqc_matrix_determinant()</code> . . . . .	54
5.1.2.47 <code>mqc_matrix_diag2full()</code> . . . . .	54
5.1.2.48 <code>mqc_matrix_diag2symm()</code> . . . . .	55
5.1.2.49 <code>mqc_matrix_diagmatrix_put_complex()</code> . . . . .	56
5.1.2.50 <code>mqc_matrix_diagmatrix_put_integer()</code> . . . . .	56
5.1.2.51 <code>mqc_matrix_diagmatrix_put_real()</code> . . . . .	57
5.1.2.52 <code>mqc_matrix_diagmatrix_put_vector()</code> . . . . .	58
5.1.2.53 <code>mqc_matrix_diagonalize()</code> . . . . .	59
5.1.2.54 <code>mqc_matrix_full2diag()</code> . . . . .	59
5.1.2.55 <code>mqc_matrix_full2symm()</code> . . . . .	60
5.1.2.56 <code>mqc_matrix_generalized_eigensystem()</code> . . . . .	61
5.1.2.57 <code>mqc_matrix_havecomplex()</code> . . . . .	61
5.1.2.58 <code>mqc_matrix_havediagonal()</code> . . . . .	61
5.1.2.59 <code>mqc_matrix_havefull()</code> . . . . .	62
5.1.2.60 <code>mqc_matrix_haveinteger()</code> . . . . .	63
5.1.2.61 <code>mqc_matrix_havereal()</code> . . . . .	63
5.1.2.62 <code>mqc_matrix_havesymmetric()</code> . . . . .	64
5.1.2.63 <code>mqc_matrix_identity()</code> . . . . .	64
5.1.2.64 <code>mqc_matrix_initialize()</code> . . . . .	65

---

5.1.2.65 <code>mqc_matrix_inverse()</code> . . . . .	65
5.1.2.66 <code>mqc_matrix_isallocated()</code> . . . . .	65
5.1.2.67 <code>mqc_matrix_matrix_at()</code> . . . . .	66
5.1.2.68 <code>mqc_matrix_matrix_contraction()</code> . . . . .	66
5.1.2.69 <code>mqc_matrix_matrix_put()</code> . . . . .	67
5.1.2.70 <code>mqc_matrix_norm()</code> . . . . .	68
5.1.2.71 <code>mqc_matrix_rms_max()</code> . . . . .	68
5.1.2.72 <code>mqc_matrix_rows()</code> . . . . .	68
5.1.2.73 <code>mqc_matrix_scalar_at()</code> . . . . .	69
5.1.2.74 <code>mqc_matrix_scalar_put()</code> . . . . .	70
5.1.2.75 <code>mqc_matrix_set()</code> . . . . .	70
5.1.2.76 <code>mqc_matrix_sqrt()</code> . . . . .	70
5.1.2.77 <code>mqc_matrix_svd()</code> . . . . .	70
5.1.2.78 <code>mqc_matrix_symm2diag()</code> . . . . .	70
5.1.2.79 <code>mqc_matrix_symm2full()</code> . . . . .	71
5.1.2.80 <code>mqc_matrix_symm2full_func()</code> . . . . .	72
5.1.2.81 <code>mqc_matrix_symmetrize()</code> . . . . .	72
5.1.2.82 <code>mqc_matrix_symmmatrix_put_complex()</code> . . . . .	73
5.1.2.83 <code>mqc_matrix_symmmatrix_put_integer()</code> . . . . .	74
5.1.2.84 <code>mqc_matrix_symmmatrix_put_real()</code> . . . . .	74
5.1.2.85 <code>mqc_matrix_symmsymmr4tensor_put_complex()</code> . . . . .	75
5.1.2.86 <code>mqc_matrix_symmsymmr4tensor_put_real()</code> . . . . .	75
5.1.2.87 <code>mqc_matrix_test_diagonal()</code> . . . . .	75
5.1.2.88 <code>mqc_matrix_test_symmetric()</code> . . . . .	76
5.1.2.89 <code>mqc_matrix_trace()</code> . . . . .	77
5.1.2.90 <code>mqc_matrix_transpose()</code> . . . . .	77
5.1.2.91 <code>mqc_matrix_vector_at()</code> . . . . .	78
5.1.2.92 <code>mqc_matrix_vector_put()</code> . . . . .	79
5.1.2.93 <code>mqc_matrixmatrixdotproduct()</code> . . . . .	80
5.1.2.94 <code>mqc_matrixmatrixproduct()</code> . . . . .	80
5.1.2.95 <code>mqc_matrixmatrixsubtract()</code> . . . . .	81
5.1.2.96 <code>mqc_matrixmatrixsum()</code> . . . . .	81
5.1.2.97 <code>mqc_matrixscalarproduct()</code> . . . . .	82
5.1.2.98 <code>mqc_matrixvectordotproduct()</code> . . . . .	82
5.1.2.99 <code>mqc_outer()</code> . . . . .	82
5.1.2.100 <code>mqc_output_complex_scalar()</code> . . . . .	83
5.1.2.101 <code>mqc_output_integer_scalar()</code> . . . . .	84
5.1.2.102 <code>mqc_output_mqcscalar_scalar()</code> . . . . .	84
5.1.2.103 <code>mqc_output_real_scalar()</code> . . . . .	85

---

5.1.2.104 mqc_print_matrix_algebra1()	86
5.1.2.105 mqc_print_r4tensor_algebra1()	87
5.1.2.106 mqc_print_scalar_algebra1()	87
5.1.2.107 mqc_print_vector_algebra1()	88
5.1.2.108 mqc_r4tensor_at()	89
5.1.2.109 mqc_r4tensor_havecomplex()	89
5.1.2.110 mqc_r4tensor_haveinteger()	89
5.1.2.111 mqc_r4tensor_havereal()	89
5.1.2.112 mqc_r4tensor_initialize()	90
5.1.2.113 mqc_r4tensor_put()	90
5.1.2.114 mqc_realgtscalar()	90
5.1.2.115 mqc_reallescalar()	91
5.1.2.116 mqc_realltscalar()	92
5.1.2.117 mqc_realscalaradd()	92
5.1.2.118 mqc_realscalddivide()	93
5.1.2.119 mqc_realscalarmultiply()	94
5.1.2.120 mqc_realscalarsubtract()	94
5.1.2.121 mqc_realvectorproduct()	95
5.1.2.122 mqc_scalar_acos()	96
5.1.2.123 mqc_scalar_asin()	96
5.1.2.124 mqc_scalar_atan()	97
5.1.2.125 mqc_scalar_atan2()	98
5.1.2.126 mqc_scalar_cmplx()	98
5.1.2.127 mqc_scalar_complex_conjugate()	99
5.1.2.128 mqc_scalar_complex_imagpart()	100
5.1.2.129 mqc_scalar_complex_realpart()	100
5.1.2.130 mqc_scalar_cos()	101
5.1.2.131 mqc_scalar_get_abs_value()	101
5.1.2.132 mqc_scalar_get_intrinsic_complex()	102
5.1.2.133 mqc_scalar_get_intrinsic_integer()	103
5.1.2.134 mqc_scalar_get_intrinsic_real()	103
5.1.2.135 mqc_scalar_get_random_value()	104
5.1.2.136 mqc_scalar_havecomplex()	105
5.1.2.137 mqc_scalar_haveinteger()	105
5.1.2.138 mqc_scalar_havereal()	106
5.1.2.139 mqc_scalar_isallocated()	107
5.1.2.140 mqc_scalar_sin()	107
5.1.2.141 mqc_scalar_sqrt()	108
5.1.2.142 mqc_scalar_tan()	109

---

---

5.1.2.143 <code>mqc_scalaradd()</code> . . . . .	109
5.1.2.144 <code>mqc_scalarcomplexadd()</code> . . . . .	110
5.1.2.145 <code>mqc_scalarcomplexdivide()</code> . . . . .	111
5.1.2.146 <code>mqc_scalarcomplexexponent()</code> . . . . .	111
5.1.2.147 <code>mqc_scalarcomplexmultiply()</code> . . . . .	112
5.1.2.148 <code>mqc_scalarcomplexsubtract()</code> . . . . .	113
5.1.2.149 <code>mqc_scalardivide()</code> . . . . .	113
5.1.2.150 <code>mqc_scalareq()</code> . . . . .	114
5.1.2.151 <code>mqc_scalarexponent()</code> . . . . .	115
5.1.2.152 <code>mqc_scalarge()</code> . . . . .	115
5.1.2.153 <code>mqc_scalargt()</code> . . . . .	116
5.1.2.154 <code>mqc_scalargtinteger()</code> . . . . .	117
5.1.2.155 <code>mqc_scalargtrealt()</code> . . . . .	118
5.1.2.156 <code>mqc_scalarintegeradd()</code> . . . . .	119
5.1.2.157 <code>mqc_scalarintegerdivide()</code> . . . . .	119
5.1.2.158 <code>mqc_scalarintegerexponent()</code> . . . . .	120
5.1.2.159 <code>mqc_scalarintegermultiply()</code> . . . . .	121
5.1.2.160 <code>mqc_scalarintegersubtract()</code> . . . . .	121
5.1.2.161 <code>mqc_scalarle()</code> . . . . .	122
5.1.2.162 <code>mqc_scalarleinteger()</code> . . . . .	123
5.1.2.163 <code>mqc_scalarlerealt()</code> . . . . .	124
5.1.2.164 <code>mqc_scalarlt()</code> . . . . .	124
5.1.2.165 <code>mqc_scalarltrealt()</code> . . . . .	125
5.1.2.166 <code>mqc_scalarmatrixproduct()</code> . . . . .	126
5.1.2.167 <code>mqc_scalarmultiply()</code> . . . . .	126
5.1.2.168 <code>mqc_scalarne()</code> . . . . .	127
5.1.2.169 <code>mqc_scalarrealadd()</code> . . . . .	127
5.1.2.170 <code>mqc_scalarrealddivide()</code> . . . . .	128
5.1.2.171 <code>mqc_scalarrealexponent()</code> . . . . .	129
5.1.2.172 <code>mqc_scalarrealmultiply()</code> . . . . .	129
5.1.2.173 <code>mqc_scalarrealsubtract()</code> . . . . .	130
5.1.2.174 <code>mqc_scalarsubtract()</code> . . . . .	131
5.1.2.175 <code>mqc_scalarvectordifference()</code> . . . . .	131
5.1.2.176 <code>mqc_scalarvectorproduct()</code> . . . . .	132
5.1.2.177 <code>mqc_scalarvectorsum()</code> . . . . .	133
5.1.2.178 <code>mqc_set_array2tensor()</code> . . . . .	134
5.1.2.179 <code>mqc_set_array2vector_complex()</code> . . . . .	134
5.1.2.180 <code>mqc_set_array2vector_integer()</code> . . . . .	135
5.1.2.181 <code>mqc_set_array2vector_real()</code> . . . . .	135

5.1.2.182 mqc_set_complexarray2matrix()	136
5.1.2.183 mqc_set_integerarray2matrix()	137
5.1.2.184 mqc_set_matrix2complexarray()	138
5.1.2.185 mqc_set_matrix2integerarray()	138
5.1.2.186 mqc_set_matrix2matrix()	139
5.1.2.187 mqc_set_matrix2realarray()	140
5.1.2.188 mqc_set_realarray2matrix()	141
5.1.2.189 mqc_set_vector2complexarray()	141
5.1.2.190 mqc_set_vector2integerarray()	142
5.1.2.191 mqc_set_vector2realarray()	143
5.1.2.192 mqc_set_vector2vector()	144
5.1.2.193 mqc_vector2diagmatrix()	144
5.1.2.194 mqc_vector_abs()	145
5.1.2.195 mqc_vector_argsort()	146
5.1.2.196 mqc_vector_cast_complex()	146
5.1.2.197 mqc_vector_cast_integer()	147
5.1.2.198 mqc_vector_cast_real()	147
5.1.2.199 mqc_vector_cmplx()	148
5.1.2.200 mqc_vector_complex_imagpart()	149
5.1.2.201 mqc_vector_complex_realpart()	149
5.1.2.202 mqc_vector_conjugate_transpose()	150
5.1.2.203 mqc_vector_copy_complex2int()	151
5.1.2.204 mqc_vector_copy_complex2real()	151
5.1.2.205 mqc_vector_copy_int2complex()	152
5.1.2.206 mqc_vector_copy_int2real()	152
5.1.2.207 mqc_vector_copy_real2complex()	153
5.1.2.208 mqc_vector_copy_real2int()	154
5.1.2.209 mqc_vector_havecomplex()	154
5.1.2.210 mqc_vector_haveinteger()	155
5.1.2.211 mqc_vector_havereal()	155
5.1.2.212 mqc_vector_initialize()	157
5.1.2.213 mqc_vector_isallocated()	158
5.1.2.214 mqc_vector_iscolumn()	158
5.1.2.215 mqc_vector_maxloc()	159
5.1.2.216 mqc_vector_maxval()	160
5.1.2.217 mqc_vector_minloc()	160
5.1.2.218 mqc_vector_minval()	161
5.1.2.219 mqc_vector_norm()	161
5.1.2.220 mqc_vector_pop()	162



5.1.2.221 mqc_vector_power()	163
5.1.2.222 mqc_vector_push()	164
5.1.2.223 mqc_vector_scalar_at()	164
5.1.2.224 mqc_vector_scalar_increment()	165
5.1.2.225 mqc_vector_scalar_put()	166
5.1.2.226 mqc_vector_shift()	167
5.1.2.227 mqc_vector_sort()	167
5.1.2.228 mqc_vector_sqrt()	168
5.1.2.229 mqc_vector_transpose()	169
5.1.2.230 mqc_vector_unshift()	169
5.1.2.231 mqc_vector_vector_at()	170
5.1.2.232 mqc_vector_vector_put()	171
5.1.2.233 mqc_vectorcomplexdivide()	171
5.1.2.234 mqc_vectorcomplexproduct()	172
5.1.2.235 mqc_vectorintegerdivide()	173
5.1.2.236 mqc_vectorintegerproduct()	174
5.1.2.237 mqc_vectormatrixdotproduct()	174
5.1.2.238 mqc_vectorrealdive()	175
5.1.2.239 mqc_vectorrealproduct()	176
5.1.2.240 mqc_vectorscalardive()	177
5.1.2.241 mqc_vectorscalarproduct()	178
5.1.2.242 mqc_vectorvectordifference()	178
5.1.2.243 mqc_vectorvectordotproduct()	179
5.1.2.244 mqc_vectorvectorsum()	180
5.1.2.245 symindexhash()	181
5.2 mqc_est Module Reference	181
5.2.1 Function/Subroutine Documentation	183
5.2.1.1 gen_det_str()	184
5.2.1.2 get_one_gamma_matrix()	184
5.2.1.3 mqc_build_ci_hamiltonian()	184
5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()	184
5.2.1.5 mqc_eigenvalues_add_name()	185
5.2.1.6 mqc_eigenvalues_allocate()	185
5.2.1.7 mqc_eigenvalues_array_name()	185
5.2.1.8 mqc_eigenvalues_array_type()	185
5.2.1.9 mqc_eigenvalues_at()	185
5.2.1.10 mqc_eigenvalues_dimension()	186
5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()	186
5.2.1.12 mqc_eigenvalues_has_alpha()	186

5.2.1.13 mqc_eigenvalues_has_beta()	186
5.2.1.14 mqc_eigenvalues_integral_multiply()	186
5.2.1.15 mqc_eigenvalues_isallocated()	186
5.2.1.16 mqc_eigenvalues_output_array()	187
5.2.1.17 mqc_eigenvalues_output_block()	187
5.2.1.18 mqc_eri_integral_contraction()	187
5.2.1.19 mqc_integral_add_name()	187
5.2.1.20 mqc_integral_allocate()	187
5.2.1.21 mqc_integral_array_name()	188
5.2.1.22 mqc_integral_array_type()	188
5.2.1.23 mqc_integral_at()	188
5.2.1.24 mqc_integral_conjugate_transpose()	188
5.2.1.25 mqc_integral_delete_energy_list()	188
5.2.1.26 mqc_integral_difference()	188
5.2.1.27 mqc_integral_dimension()	189
5.2.1.28 mqc_integral_eigenvalues_multiply()	189
5.2.1.29 mqc_integral_get_energy_list()	189
5.2.1.30 mqc_integral_has_alpha()	189
5.2.1.31 mqc_integral_has_alphabeta()	189
5.2.1.32 mqc_integral_has_beta()	189
5.2.1.33 mqc_integral_has_betaalpha()	190
5.2.1.34 mqc_integral_identity()	190
5.2.1.35 mqc_integral_initialize()	190
5.2.1.36 mqc_integral_integral_multiply()	190
5.2.1.37 mqc_integral_isallocated()	190
5.2.1.38 mqc_integral_matrix_multiply()	191
5.2.1.39 mqc_integral_norm()	191
5.2.1.40 mqc_integral_output_array()	191
5.2.1.41 mqc_integral_output_block()	191
5.2.1.42 mqc_integral_output_orbitals()	191
5.2.1.43 mqc_integral_scalar_multiply()	192
5.2.1.44 mqc_integral_set_energy_list()	192
5.2.1.45 mqc_integral_sum()	192
5.2.1.46 mqc_integral_swap_orbitals()	192
5.2.1.47 mqc_integral_transpose()	192
5.2.1.48 mqc_matrix_integral_multiply()	193
5.2.1.49 mqc_matrix_spinblockghf()	193
5.2.1.50 mqc_matrix_undospinblockghf_eigenvalues()	193
5.2.1.51 mqc_matrix_undospinblockghf_integral()	193

5.2.1.52 mqc_print_eigenvalues()	193
5.2.1.53 mqc_print_integral()	194
5.2.1.54 mqc_print_twoeris()	194
5.2.1.55 mqc_print_wavefunction()	194
5.2.1.56 mqc_scalar_integral_multiply()	194
5.2.1.57 mqc_scf_eigenvalues_power()	194
5.2.1.58 mqc_scf_integral_contraction()	195
5.2.1.59 mqc_scf_integral_determinant()	195
5.2.1.60 mqc_scf_integral_diagonalize()	195
5.2.1.61 mqc_scf_integral_generalized_eigensystem()	195
5.2.1.62 mqc_scf_integral_inverse()	195
5.2.1.63 mqc_scf_integral_trace()	195
5.2.1.64 mqc_scf_transformation_matrix()	196
5.2.1.65 mqc_twoeris_allocate()	196
5.2.1.66 mqc_twoeris_at()	196
5.2.1.67 slater_condon()	196
5.2.1.68 twoeri_trans()	197

## 6 Data Type Documentation 199

6.1 mqc_algebra::abs Interface Reference	199
6.1.1 Detailed Description	199
6.1.2 Member Function/Subroutine Documentation	199
6.1.2.1 mqc_scalar_get_abs_value()	199
6.1.2.2 mqc_vector_abs()	200
6.2 mqc_algebra::acos Interface Reference	201
6.2.1 Detailed Description	201
6.2.2 Member Function/Subroutine Documentation	201
6.2.2.1 mqc_scalar_acos()	201
6.3 mqc_algebra::aimag Interface Reference	202
6.3.1 Detailed Description	202
6.3.2 Member Function/Subroutine Documentation	202
6.3.2.1 mqc_scalar_complex_imagpart()	202
6.3.2.2 mqc_vector_complex_imagpart()	203
6.4 mqc_algebra::asin Interface Reference	204
6.4.1 Detailed Description	204
6.4.2 Member Function/Subroutine Documentation	204
6.4.2.1 mqc_scalar_asin()	204
6.5 mqc_algebra::assignment(=) Interface Reference	205
6.5.1 Detailed Description	206

6.5.2 Member Function/Subroutine Documentation	206
6.5.2.1 mqc_input_complex_scalar()	206
6.5.2.2 mqc_input_integer_scalar()	207
6.5.2.3 mqc_input_real_scalar()	207
6.5.2.4 mqc_output_complex_scalar()	208
6.5.2.5 mqc_output_integer_scalar()	209
6.5.2.6 mqc_output_mqcscalar_scalar()	209
6.5.2.7 mqc_output_real_scalar()	210
6.5.2.8 mqc_set_array2tensor()	211
6.5.2.9 mqc_set_array2vector_complex()	211
6.5.2.10 mqc_set_array2vector_integer()	212
6.5.2.11 mqc_set_array2vector_real()	212
6.5.2.12 mqc_set_complexarray2matrix()	213
6.5.2.13 mqc_set_integerarray2matrix()	214
6.5.2.14 mqc_set_matrix2complexarray()	215
6.5.2.15 mqc_set_matrix2integerarray()	215
6.5.2.16 mqc_set_matrix2matrix()	216
6.5.2.17 mqc_set_matrix2realarray()	217
6.5.2.18 mqc_set_realarray2matrix()	218
6.5.2.19 mqc_set_vector2complexarray()	218
6.5.2.20 mqc_set_vector2integerarray()	219
6.5.2.21 mqc_set_vector2realarray()	220
6.5.2.22 mqc_set_vector2vector()	221
6.6 mqc_est::assignment(=) Interface Reference	221
6.6.1 Member Function/Subroutine Documentation	222
6.6.1.1 mqc_eigenvalues_output_array()	222
6.6.1.2 mqc_integral_output_array()	222
6.7 mqc_algebra::atan Interface Reference	222
6.7.1 Detailed Description	222
6.7.2 Member Function/Subroutine Documentation	222
6.7.2.1 mqc_scalar_atan()	222
6.8 mqc_algebra::atan2 Interface Reference	223
6.8.1 Detailed Description	223
6.8.2 Member Function/Subroutine Documentation	223
6.8.2.1 mqc_scalar_atan2()	223
6.9 mqc_algebra::cmplx Interface Reference	224
6.9.1 Detailed Description	224
6.9.2 Member Function/Subroutine Documentation	224
6.9.2.1 mqc_scalar_cmplx()	224

6.9.2.2 <code>mqc_vector_cmplx()</code>	225
6.10 <code>mqc_algebra::conjg</code> Interface Reference	226
6.10.1 Detailed Description	226
6.10.2 Member Function/Subroutine Documentation	226
6.10.2.1 <code>mqc_scalar_complex_conjugate()</code>	226
6.11 <code>mqc_algebra::contraction</code> Interface Reference	227
6.11.1 Detailed Description	227
6.11.2 Member Function/Subroutine Documentation	227
6.11.2.1 <code>mqc_matrix_matrix_contraction()</code>	227
6.12 <code>mqc_est::contraction</code> Interface Reference	228
6.12.1 Member Function/Subroutine Documentation	228
6.12.1.1 <code>mqc_eri_integral_contraction()</code>	228
6.12.1.2 <code>mqc_scf_integral_contraction()</code>	228
6.13 <code>mqc_algebra::cos</code> Interface Reference	228
6.13.1 Detailed Description	228
6.13.2 Member Function/Subroutine Documentation	229
6.13.2.1 <code>mqc_scalar_cos()</code>	229
6.14 <code>mqc_algebra::dagger</code> Interface Reference	229
6.14.1 Detailed Description	230
6.14.2 Member Function/Subroutine Documentation	230
6.14.2.1 <code>mqc_matrix_conjugate_transpose()</code>	230
6.14.2.2 <code>mqc_vector_conjugate_transpose()</code>	230
6.15 <code>mqc_est::dagger</code> Interface Reference	231
6.15.1 Member Function/Subroutine Documentation	231
6.15.1.1 <code>mqc_integral_conjugate_transpose()</code>	231
6.16 <code>mqc_algebra::dot_product</code> Interface Reference	231
6.16.1 Detailed Description	232
6.16.2 Member Function/Subroutine Documentation	232
6.16.2.1 <code>mqc_vectorvectordotproduct()</code>	232
6.17 <code>mqc_est::dot_product</code> Interface Reference	233
6.17.1 Member Function/Subroutine Documentation	233
6.17.1.1 <code>mqc_eigenvalue_eigenvalue_dotproduct()</code>	233
6.18 <code>mqc_algebra::matmul</code> Interface Reference	233
6.18.1 Detailed Description	233
6.18.2 Member Function/Subroutine Documentation	234
6.18.2.1 <code>mqc_matrixmatrixdotproduct()</code>	234
6.18.2.2 <code>mqc_matrixvectordotproduct()</code>	234
6.18.2.3 <code>mqc_vectormatrixdotproduct()</code>	234
6.19 <code>mqc_est::matmul</code> Interface Reference	234

6.19.1 Member Function/Subroutine Documentation	234
6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()	235
6.19.1.2 mqc_eigenvalues_integral_multiply()	235
6.19.1.3 mqc_integral_eigenvalues_multiply()	235
6.19.1.4 mqc_integral_integral_multiply()	235
6.19.1.5 mqc_integral_matrix_multiply()	235
6.19.1.6 mqc_matrix_integral_multiply()	236
6.20 mqc_algebra::matrix_symm2sq Interface Reference	236
6.20.1 Detailed Description	236
6.20.2 Member Function/Subroutine Documentation	236
6.20.2.1 matrix_symm2sq_complex()	236
6.20.2.2 matrix_symm2sq_integer()	237
6.20.2.3 matrix_symm2sq_real()	238
6.21 mqc_algebra::mqc_cast_complex Interface Reference	239
6.21.1 Detailed Description	239
6.21.2 Member Function/Subroutine Documentation	239
6.21.2.1 mqc_matrix_cast_complex()	239
6.21.2.2 mqc_vector_cast_complex()	240
6.22 mqc_algebra::mqc_cast_integer Interface Reference	241
6.22.1 Detailed Description	241
6.22.2 Member Function/Subroutine Documentation	241
6.22.2.1 mqc_matrix_cast_integer()	241
6.22.2.2 mqc_vector_cast_integer()	242
6.23 mqc_algebra::mqc_cast_real Interface Reference	242
6.23.1 Detailed Description	243
6.23.2 Member Function/Subroutine Documentation	243
6.23.2.1 mqc_matrix_cast_real()	243
6.23.2.2 mqc_vector_cast_real()	243
6.24 mqc_est::mqc_determinant Type Reference	244
6.24.1 Member Data Documentation	244
6.24.1.1 nalpstr	244
6.24.1.2 nbetstr	244
6.24.1.3 ndets	245
6.24.1.4 order	245
6.24.1.5 strings	245
6.25 mqc_est::mqc_determinant_string Type Reference	245
6.25.1 Member Data Documentation	245
6.25.1.1 alpha	245
6.25.1.2 beta	245

6.26 <code>mqc_algebra::mqc_have_complex</code> Interface Reference	246
6.26.1 Detailed Description	246
6.26.2 Member Function/Subroutine Documentation	246
6.26.2.1 <code>mqc_matrix_havecomplex()</code>	246
6.26.2.2 <code>mqc_vector_havecomplex()</code>	247
6.27 <code>mqc_algebra::mqc_have_int</code> Interface Reference	247
6.27.1 Detailed Description	248
6.27.2 Member Function/Subroutine Documentation	248
6.27.2.1 <code>mqc_matrix_haveinteger()</code>	248
6.27.2.2 <code>mqc_vector_haveinteger()</code>	249
6.28 <code>mqc_algebra::mqc_have_real</code> Interface Reference	249
6.28.1 Detailed Description	250
6.28.2 Member Function/Subroutine Documentation	250
6.28.2.1 <code>mqc_matrix_havereal()</code>	250
6.28.2.2 <code>mqc_vector_havereal()</code>	250
6.29 <code>mqc_algebra::mqc_matrix</code> Type Reference	251
6.29.1 Detailed Description	252
6.29.2 Member Function/Subroutine Documentation	252
6.29.2.1 <code>at()</code>	252
6.29.2.2 <code>dagger()</code>	252
6.29.2.3 <code>det()</code>	253
6.29.2.4 <code>diag()</code>	253
6.29.2.5 <code>eigensys()</code>	253
6.29.2.6 <code>identity()</code>	253
6.29.2.7 <code>init()</code>	253
6.29.2.8 <code>inv()</code>	253
6.29.2.9 <code>mat()</code>	254
6.29.2.10 <code>mput()</code>	254
6.29.2.11 <code>norm()</code>	254
6.29.2.12 <code>print()</code>	254
6.29.2.13 <code>put()</code>	254
6.29.2.14 <code>rmsmax()</code>	254
6.29.2.15 <code>set()</code>	255
6.29.2.16 <code>sqr()</code>	255
6.29.2.17 <code>svd()</code>	255
6.29.2.18 <code>trace()</code>	255
6.29.2.19 <code>transpose()</code>	255
6.29.2.20 <code>vat()</code>	255
6.29.2.21 <code>vput()</code>	256

6.30 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference	256
6.30.1 Detailed Description	256
6.30.2 Member Function/Subroutine Documentation	256
6.30.2.1 mqc_matrix_diagmatrix_put_complex()	256
6.30.2.2 mqc_matrix_diagmatrix_put_integer()	257
6.30.2.3 mqc_matrix_diagmatrix_put_real()	258
6.30.2.4 mqc_matrix_diagmatrix_put_vector()	259
6.31 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference	259
6.31.1 Detailed Description	260
6.31.2 Member Function/Subroutine Documentation	260
6.31.2.1 mqc_matrix_symmmatrix_put_complex()	260
6.31.2.2 mqc_matrix_symmmatrix_put_integer()	261
6.31.2.3 mqc_matrix_symmmatrix_put_real()	261
6.32 mqc_est::mqc_matrix_undospinblockghf Interface Reference	262
6.32.1 Member Function/Subroutine Documentation	262
6.32.1.1 mqc_matrix_undospinblockghf_eigenvalues()	262
6.32.1.2 mqc_matrix_undospinblockghf_integral()	263
6.33 mqc_algebra::mqc_print Interface Reference	263
6.33.1 Detailed Description	263
6.33.2 Member Function/Subroutine Documentation	263
6.33.2.1 mqc_print_matrix_algebra1()	263
6.33.2.2 mqc_print_r4tensor_algebra1()	264
6.33.2.3 mqc_print_scalar_algebra1()	265
6.33.2.4 mqc_print_vector_algebra1()	266
6.34 mqc_est::mqc_print Interface Reference	267
6.34.1 Member Function/Subroutine Documentation	267
6.34.1.1 mqc_print_eigenvalues()	267
6.34.1.2 mqc_print_integral()	267
6.34.1.3 mqc_print_twoeris()	268
6.34.1.4 mqc_print_wavefunction()	268
6.35 mqc_est::mqc_pscf_wavefunction Type Reference	268
6.35.1 Member Data Documentation	268
6.35.1.1 nactive	269
6.35.1.2 ncore	269
6.35.1.3 nfrz	269
6.35.1.4 nval	269
6.35.1.5 pscf_amplitudes	269
6.35.1.6 pscf_energies	269
6.36 mqc_algebra::mqc_r4tensor Type Reference	269



6.36.1 Detailed Description	270
6.36.2 Member Function/Subroutine Documentation	270
6.36.2.1 at()	270
6.36.2.2 init()	270
6.36.2.3 print()	270
6.36.2.4 put()	271
6.37 mqc_algebra::mqc_scalar Type Reference	271
6.37.1 Detailed Description	271
6.37.2 Member Function/Subroutine Documentation	271
6.37.2.1 abs()	271
6.37.2.2 cval()	272
6.37.2.3 ival()	272
6.37.2.4 print()	272
6.37.2.5 random()	272
6.37.2.6 rval()	272
6.38 mqc_est::mqc_scf_eigenvalues Type Reference	273
6.38.1 Member Function/Subroutine Documentation	273
6.38.1.1 addlabel()	273
6.38.1.2 at()	273
6.38.1.3 getblock()	273
6.38.1.4 getlabel()	273
6.38.1.5 power()	273
6.38.1.6 print()	274
6.39 mqc_est::mqc_scf_integral Type Reference	274
6.39.1 Member Function/Subroutine Documentation	274
6.39.1.1 addlabel()	274
6.39.1.2 deleteelist()	274
6.39.1.3 det()	275
6.39.1.4 diag()	275
6.39.1.5 eigensys()	275
6.39.1.6 getblock()	275
6.39.1.7 getelist()	275
6.39.1.8 getlabel()	275
6.39.1.9 identity()	275
6.39.1.10 init()	276
6.39.1.11 inv()	276
6.39.1.12 norm()	276
6.39.1.13 orbitals()	276
6.39.1.14 print()	276

6.39.1.15 setelist()	276
6.39.1.16 swap()	276
6.39.1.17 trace()	277
6.40 mqc_algebra::mqc_set_array2vector Interface Reference	277
6.40.1 Detailed Description	277
6.40.2 Member Function/Subroutine Documentation	277
6.40.2.1 mqc_set_array2vector_complex()	277
6.40.2.2 mqc_set_array2vector_integer()	278
6.40.2.3 mqc_set_array2vector_real()	279
6.41 mqc_est::mqc_twoeris Type Reference	280
6.41.1 Member Function/Subroutine Documentation	280
6.41.1.1 print()	280
6.42 mqc_algebra::mqc_vector Type Reference	280
6.42.1 Detailed Description	281
6.42.2 Member Function/Subroutine Documentation	281
6.42.2.1 abs()	281
6.42.2.2 argsort()	282
6.42.2.3 at()	282
6.42.2.4 dagger()	282
6.42.2.5 diag()	282
6.42.2.6 init()	282
6.42.2.7 maxloc()	282
6.42.2.8 maxval()	283
6.42.2.9 minloc()	283
6.42.2.10 minval()	283
6.42.2.11 norm()	283
6.42.2.12 pop()	283
6.42.2.13 power()	283
6.42.2.14 print()	284
6.42.2.15 push()	284
6.42.2.16 put()	284
6.42.2.17 shift()	284
6.42.2.18 size()	284
6.42.2.19 sort()	284
6.42.2.20 sqrt()	285
6.42.2.21 transpose()	285
6.42.2.22 unshift()	285
6.42.2.23 vat()	285
6.42.2.24 vput()	285

6.43 mqc_est::mqc_wavefunction Type Reference	286
6.43.1 Member Function/Subroutine Documentation	286
6.43.1.1 print()	286
6.43.2 Member Data Documentation	287
6.43.2.1 basis	287
6.43.2.2 charge	287
6.43.2.3 core_hamiltonian	287
6.43.2.4 density_matrix	287
6.43.2.5 fock_matrix	287
6.43.2.6 mo_coefficients	287
6.43.2.7 mo_energies	288
6.43.2.8 mo_symmetries	288
6.43.2.9 multiplicity	288
6.43.2.10 nalpha	288
6.43.2.11 nbasis	288
6.43.2.12 nbeta	288
6.43.2.13 nelelectrons	288
6.43.2.14 overlap_matrix	289
6.43.2.15 scf_density_matrix	289
6.43.2.16 symmetry	289
6.43.2.17 wf_complex	289
6.43.2.18 wf_type	289
6.44 mqc_algebra::operator(*) Interface Reference	289
6.44.1 Detailed Description	290
6.44.2 Member Function/Subroutine Documentation	291
6.44.2.1 mqc_complexscalarmultiply()	291
6.44.2.2 mqc_complexvectorproduct()	291
6.44.2.3 mqc_integerscalarmultiply()	292
6.44.2.4 mqc_integervectorproduct()	293
6.44.2.5 mqc_matrixmatrixproduct()	293
6.44.2.6 mqc_matrixscalarproduct()	294
6.44.2.7 mqc_realscalarmultiply()	294
6.44.2.8 mqc_realvectorproduct()	295
6.44.2.9 mqc_scalarcomplexmultiply()	296
6.44.2.10 mqc_scalarintegermultiply()	297
6.44.2.11 mqc_scalarmatrixproduct()	297
6.44.2.12 mqc_scalarmultiply()	297
6.44.2.13 mqc_scalarrealmultiply()	298
6.44.2.14 mqc_scalarvectorproduct()	299

6.44.2.15 <code>mqc_vectorcomplexproduct()</code>	299
6.44.2.16 <code>mqc_vectorintegerproduct()</code>	300
6.44.2.17 <code>mqc_vectorrealproduct()</code>	301
6.44.2.18 <code>mqc_vectorscalarproduct()</code>	302
6.45 <code>mqc_est::operator(*)</code> Interface Reference	302
6.45.1 Member Function/Subroutine Documentation	302
6.45.1.1 <code>mqc_integral_scalar_multiply()</code>	303
6.45.1.2 <code>mqc_scalar_integral_multiply()</code>	303
6.46 <code>mqc_algebra::operator(**)</code> Interface Reference	303
6.46.1 Detailed Description	303
6.46.2 Member Function/Subroutine Documentation	303
6.46.2.1 <code>mqc_scalarcomplexexponent()</code>	304
6.46.2.2 <code>mqc_scalarexponent()</code>	304
6.46.2.3 <code>mqc_scalarintegerexponent()</code>	305
6.46.2.4 <code>mqc_scalarrealexponent()</code>	306
6.47 <code>mqc_algebra::operator(+)</code> Interface Reference	306
6.47.1 Detailed Description	307
6.47.2 Member Function/Subroutine Documentation	307
6.47.2.1 <code>mqc_complexscalaradd()</code>	307
6.47.2.2 <code>mqc_integerscalaradd()</code>	308
6.47.2.3 <code>mqc_matrixmatrixsum()</code>	309
6.47.2.4 <code>mqc_realscalaradd()</code>	309
6.47.2.5 <code>mqc_scalaradd()</code>	310
6.47.2.6 <code>mqc_scalarcomplexadd()</code>	311
6.47.2.7 <code>mqc_scalarintegeradd()</code>	311
6.47.2.8 <code>mqc_scalarrealadd()</code>	312
6.47.2.9 <code>mqc_scalarvectorsum()</code>	313
6.47.2.10 <code>mqc_vectorvectorsum()</code>	313
6.48 <code>mqc_est::operator(+)</code> Interface Reference	314
6.48.1 Member Function/Subroutine Documentation	314
6.48.1.1 <code>mqc_integral_sum()</code>	314
6.49 <code>mqc_est::operator(-)</code> Interface Reference	315
6.49.1 Member Function/Subroutine Documentation	315
6.49.1.1 <code>mqc_integral_difference()</code>	315
6.50 <code>mqc_algebra::operator(-)</code> Interface Reference	315
6.50.1 Detailed Description	316
6.50.2 Member Function/Subroutine Documentation	316
6.50.2.1 <code>mqc_complexscalarsubtract()</code>	316
6.50.2.2 <code>mqc_integerscalarsubtract()</code>	316

6.50.2.3 mqc_matrixmatrixsubtract()	317
6.50.2.4 mqc_realscalarsubtract()	318
6.50.2.5 mqc_scalarcomplexsubtract()	318
6.50.2.6 mqc_scalarintegersubtract()	319
6.50.2.7 mqc_scalarrealsubtract()	320
6.50.2.8 mqc_scalarsubtract()	321
6.50.2.9 mqc_scalarvectordifference()	321
6.50.2.10 mqc_vectorvectordifference()	322
6.51 mqc_algebra::operator(.dot.) Interface Reference	323
6.51.1 Detailed Description	323
6.51.2 Member Function/Subroutine Documentation	323
6.51.2.1 mqc_matrixmatrixdotproduct()	323
6.51.2.2 mqc_matrixvectordotproduct()	324
6.51.2.3 mqc_vectormatrixdotproduct()	324
6.51.2.4 mqc_vectorvectordotproduct()	324
6.52 mqc_algebra::operator(.eq.) Interface Reference	325
6.52.1 Detailed Description	325
6.52.2 Member Function/Subroutine Documentation	325
6.52.2.1 mqc_scalareq()	325
6.53 mqc_algebra::operator(.ewd.) Interface Reference	326
6.53.1 Detailed Description	326
6.53.2 Member Function/Subroutine Documentation	326
6.53.2.1 mqc_elementmatrixdivide()	326
6.54 mqc_algebra::operator(.ewp.) Interface Reference	327
6.54.1 Detailed Description	327
6.54.2 Member Function/Subroutine Documentation	327
6.54.2.1 mqc_elementmatrixproduct()	328
6.54.2.2 mqc_elementvectorproduct()	328
6.55 mqc_algebra::operator(.ge.) Interface Reference	329
6.55.1 Detailed Description	329
6.55.2 Member Function/Subroutine Documentation	329
6.55.2.1 mqc_scalarge()	330
6.56 mqc_algebra::operator(.gt.) Interface Reference	330
6.56.1 Detailed Description	331
6.56.2 Member Function/Subroutine Documentation	331
6.56.2.1 mqc_integertgtscalar()	331
6.56.2.2 mqc_realtgtscalar()	332
6.56.2.3 mqc_scalargt()	333
6.56.2.4 mqc_scalargtinteger()	333

6.56.2.5 <code>mqc_scalargtreal()</code>	334
6.57 <code>mqc_algebra::operator(.le.)</code> Interface Reference	335
6.57.1 Detailed Description	335
6.57.2 Member Function/Subroutine Documentation	336
6.57.2.1 <code>mqc_integerlescalar()</code>	336
6.57.2.2 <code>mqc_reallescalar()</code>	336
6.57.2.3 <code>mqc_scalarle()</code>	337
6.57.2.4 <code>mqc_scalarleinteger()</code>	338
6.57.2.5 <code>mqc_scalarlereal()</code>	339
6.58 <code>mqc_algebra::operator(.lt.)</code> Interface Reference	340
6.58.1 Detailed Description	340
6.58.2 Member Function/Subroutine Documentation	340
6.58.2.1 <code>mqc_realltscalar()</code>	340
6.58.2.2 <code>mqc_scalarlt()</code>	341
6.58.2.3 <code>mqc_scalarltreal()</code>	342
6.59 <code>mqc_algebra::operator(.ne.)</code> Interface Reference	342
6.59.1 Detailed Description	343
6.59.2 Member Function/Subroutine Documentation	343
6.59.2.1 <code>mqc_scalarne()</code>	343
6.60 <code>mqc_algebra::operator(.outer.)</code> Interface Reference	344
6.60.1 Detailed Description	344
6.60.2 Member Function/Subroutine Documentation	344
6.60.2.1 <code>mqc_outer()</code>	344
6.61 <code>mqc_algebra::operator(.x.)</code> Interface Reference	345
6.61.1 Detailed Description	345
6.61.2 Member Function/Subroutine Documentation	345
6.61.2.1 <code>mqc_crossproduct()</code>	345
6.62 <code>mqc_algebra::operator(/)</code> Interface Reference	346
6.62.1 Detailed Description	347
6.62.2 Member Function/Subroutine Documentation	347
6.62.2.1 <code>mqc_complexscalardivide()</code>	347
6.62.2.2 <code>mqc_integerscalardivide()</code>	347
6.62.2.3 <code>mqc_realscalardivide()</code>	348
6.62.2.4 <code>mqc_scalarcomplexdivide()</code>	349
6.62.2.5 <code>mqc_scalardivide()</code>	350
6.62.2.6 <code>mqc_scalarintegerdivide()</code>	350
6.62.2.7 <code>mqc_scalarrealdivide()</code>	351
6.62.2.8 <code>mqc_vectorcomplexdivide()</code>	352
6.62.2.9 <code>mqc_vectorintegerdivide()</code>	352

6.62.2.10 <code>mqc_vectorrealdive()</code> . . . . .	353
6.62.2.11 <code>mqc_vectorscalardive()</code> . . . . .	354
6.63 <code>mqc_algebra::real</code> Interface Reference . . . . .	354
6.63.1 Detailed Description . . . . .	355
6.63.2 Member Function/Subroutine Documentation . . . . .	355
6.63.2.1 <code>mqc_scalar_complex_realpart()</code> . . . . .	355
6.63.2.2 <code>mqc_vector_complex_realpart()</code> . . . . .	356
6.64 <code>mqc_algebra::sin</code> Interface Reference . . . . .	356
6.64.1 Detailed Description . . . . .	356
6.64.2 Member Function/Subroutine Documentation . . . . .	357
6.64.2.1 <code>mqc_scalar_sin()</code> . . . . .	357
6.65 <code>mqc_algebra::sqrt</code> Interface Reference . . . . .	357
6.65.1 Detailed Description . . . . .	358
6.65.2 Member Function/Subroutine Documentation . . . . .	358
6.65.2.1 <code>mqc_scalar_sqrt()</code> . . . . .	358
6.66 <code>mqc_algebra::tan</code> Interface Reference . . . . .	358
6.66.1 Detailed Description . . . . .	359
6.66.2 Member Function/Subroutine Documentation . . . . .	359
6.66.2.1 <code>mqc_scalar_tan()</code> . . . . .	359
6.67 <code>mqc_est::transpose</code> Interface Reference . . . . .	360
6.67.1 Member Function/Subroutine Documentation . . . . .	360
6.67.1.1 <code>mqc_integral_transpose()</code> . . . . .	360
6.68 <code>mqc_algebra::transpose</code> Interface Reference . . . . .	360
6.68.1 Detailed Description . . . . .	360
6.68.2 Member Function/Subroutine Documentation . . . . .	360
6.68.2.1 <code>mqc_matrix_transpose()</code> . . . . .	360
6.68.2.2 <code>mqc_vector_transpose()</code> . . . . .	361
<b>7 File Documentation</b> . . . . .	<b>363</b>
7.1 <code>src/mqc_algebra.F03</code> File Reference . . . . .	363
7.2 <code>src/mqc_est.F03</code> File Reference . . . . .	376
<b>Index</b> . . . . .	<b>379</b>





# Chapter 1

## Modules Index

### 1.1 Modules List

Here is a list of all modules with brief descriptions:

<a href="#">mqc_algebra</a>	<b>MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran</b>	<a href="#">11</a>
<a href="#">mqc_est</a>		<a href="#">181</a>



## Chapter 2

# Data Type Index

### 2.1 Class Hierarchy

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

<code>mqc_algebra::abs</code>	199
<code>mqc_algebra::acos</code>	201
<code>mqc_algebra::aimag</code>	202
<code>mqc_algebra::asin</code>	204
<code>mqc_algebra::assignment(=)</code>	205
<code>mqc_est::assignment(=)</code>	221
<code>mqc_algebra::atan</code>	222
<code>mqc_algebra::atan2</code>	223
<code>mqc_algebra::cmplx</code>	224
<code>mqc_algebra::conjg</code>	226
<code>mqc_algebra::contraction</code>	227
<code>mqc_est::contraction</code>	228
<code>mqc_algebra::cos</code>	228
<code>mqc_algebra::dagger</code>	229
<code>mqc_est::dagger</code>	231
<code>mqc_algebra::dot_product</code>	231
<code>mqc_est::dot_product</code>	233
<code>mqc_algebra::matmul</code>	233
<code>mqc_est::matmul</code>	234
<code>mqc_algebra::matrix_symm2sq</code>	236
<code>mqc_algebra::mqc_cast_complex</code>	239
<code>mqc_algebra::mqc_cast_integer</code>	241
<code>mqc_algebra::mqc_cast_real</code>	242
<code>mqc_est::mqc_determinant</code>	244
<code>mqc_est::mqc_determinant_string</code>	245
<code>mqc_algebra::mqc_have_complex</code>	246
<code>mqc_algebra::mqc_have_int</code>	247
<code>mqc_algebra::mqc_have_real</code>	249
<code>mqc_algebra::mqc_matrix</code>	251
<code>mqc_algebra::mqc_matrix_diagmatrix_put</code>	256
<code>mqc_algebra::mqc_matrix_symmmatrix_put</code>	259

mqc_est::mqc_matrix_undospinblockghf . . . . .	262
mqc_algebra::mqc_print . . . . .	263
mqc_est::mqc_print . . . . .	267
mqc_algebra::mqc_r4tensor . . . . .	269
mqc_algebra::mqc_scalar . . . . .	271
mqc_est::mqc_scf_eigenvalues . . . . .	273
mqc_est::mqc_scf_integral . . . . .	274
mqc_algebra::mqc_set_array2vector . . . . .	277
mqc_est::mqc_twoeris . . . . .	280
mqc_algebra::mqc_vector . . . . .	280
mqc_est::mqc_wavefunction . . . . .	286
mqc_est::mqc_pscf_wavefunction . . . . .	268
mqc_algebra::operator(*) . . . . .	289
mqc_est::operator(*) . . . . .	302
mqc_algebra::operator(**) . . . . .	303
mqc_algebra::operator(+) . . . . .	306
mqc_est::operator(+) . . . . .	314
mqc_est::operator(-) . . . . .	315
mqc_algebra::operator(-) . . . . .	315
mqc_algebra::operator(.dot.) . . . . .	323
mqc_algebra::operator(.eq.) . . . . .	325
mqc_algebra::operator(.ewd.) . . . . .	326
mqc_algebra::operator(.ewp.) . . . . .	327
mqc_algebra::operator(.ge.) . . . . .	329
mqc_algebra::operator(.gt.) . . . . .	330
mqc_algebra::operator(.le.) . . . . .	335
mqc_algebra::operator(.lt.) . . . . .	340
mqc_algebra::operator(.ne.) . . . . .	342
mqc_algebra::operator(.outer.) . . . . .	344
mqc_algebra::operator(.x.) . . . . .	345
mqc_algebra::operator(/) . . . . .	346
mqc_algebra::real . . . . .	354
mqc_algebra::sin . . . . .	356
mqc_algebra::sqrt . . . . .	357
mqc_algebra::tan . . . . .	358
mqc_est::transpose . . . . .	360
mqc_algebra::transpose . . . . .	360

## Chapter 3

# Data Type Index

### 3.1 Data Types List

Here are the data types with brief descriptions:

<a href="#">mqc_algebra::abs</a>	<b>Takes the absolute value</b>	199
<a href="#">mqc_algebra::acos</a>	<b>Returns the arccosine</b>	201
<a href="#">mqc_algebra::aimag</a>	<b>Returns the imaginary part</b>	202
<a href="#">mqc_algebra::asin</a>	<b>Returns the arcsine</b>	204
<a href="#">mqc_algebra::assignment(=)</a>	<b>Assigns a variable to the value of another</b>	205
<a href="#">mqc_est::assignment(=)</a>		221
<a href="#">mqc_algebra::atan</a>	<b>Returns the arctangent</b>	222
<a href="#">mqc_algebra::atan2</a>	<b>Returns the arctangent accounting for circle quadrant</b>	223
<a href="#">mqc_algebra::cplx</a>	<b>Defines a complex number</b>	224
<a href="#">mqc_algebra::conjg</a>	<b>Returns the complex conjugate</b>	226
<a href="#">mqc_algebra::contraction</a>	<b>Contracts two arrays</b>	227
<a href="#">mqc_est::contraction</a>		228
<a href="#">mqc_algebra::cos</a>	<b>Returns the cosine</b>	228
<a href="#">mqc_algebra::dagger</a>	<b>Returns the Hermitian conjugate</b>	229
<a href="#">mqc_est::dagger</a>		231
<a href="#">mqc_algebra::dot_product</a>	<b>Returns the dot product</b>	231
<a href="#">mqc_est::dot_product</a>		233
<a href="#">mqc_algebra::matmul</a>	<b>Multiplies two arrays</b>	233

<a href="#">mqc_est::matmul</a>	234
<a href="#">mqc_algebra::matrix_symm2sq</a>	
<b>Sets a symmetric packed intrinsic array as a square packed intrinsic array</b>	236
<a href="#">mqc_algebra::mqc_cast_complex</a>	
<b>Sets an array to complex type</b>	239
<a href="#">mqc_algebra::mqc_cast_integer</a>	
<b>Sets an array to integer type</b>	241
<a href="#">mqc_algebra::mqc_cast_real</a>	
<b>Sets an array to real type</b>	242
<a href="#">mqc_est::mqc_determinant</a>	244
<a href="#">mqc_est::mqc_determinant_string</a>	245
<a href="#">mqc_algebra::mqc_have_complex</a>	
<b>Determines in an array is complex type</b>	246
<a href="#">mqc_algebra::mqc_have_int</a>	
<b>Determines in an array is integer type</b>	247
<a href="#">mqc_algebra::mqc_have_real</a>	
<b>Determines in an array is real type</b>	249
<a href="#">mqc_algebra::mqc_matrix</a>	
<b>Rank 2 array variable</b>	251
<a href="#">mqc_algebra::mqc_matrix_diagmatrix_put</a>	
<b>Sets a diagonal packed intrinsic array as an MQC Matrix object</b>	256
<a href="#">mqc_algebra::mqc_matrix_symmmatrix_put</a>	
<b>Sets a symmetric packed intrinsic array as an MQC Matrix object</b>	259
<a href="#">mqc_est::mqc_matrix_undospinblockghf</a>	262
<a href="#">mqc_algebra::mqc_print</a>	
<b>Prints an object</b>	263
<a href="#">mqc_est::mqc_print</a>	267
<a href="#">mqc_est::mqc_pscf_wavefunction</a>	268
<a href="#">mqc_algebra::mqc_r4tensor</a>	
<b>Updates the specified element of the MQC Matrix to the specified value</b>	269
<a href="#">mqc_algebra::mqc_scalar</a>	
<b>Rank 0 array variable</b>	271
<a href="#">mqc_est::mqc_scf_eigenvalues</a>	273
<a href="#">mqc_est::mqc_scf_integral</a>	274
<a href="#">mqc_algebra::mqc_set_array2vector</a>	
<b>Sets an intrinsic array as an MQC Algebra object</b>	277
<a href="#">mqc_est::mqc_twoeris</a>	280
<a href="#">mqc_algebra::mqc_vector</a>	
<b>Rank 1 array variable</b>	280
<a href="#">mqc_est::mqc_wavefunction</a>	286
<a href="#">mqc_algebra::operator(*)</a>	
<b>Multiplies two variables</b>	289
<a href="#">mqc_est::operator(*)</a>	302
<a href="#">mqc_algebra::operator(**)</a>	
<b>Exponentials a variable to the power of another</b>	303
<a href="#">mqc_algebra::operator(+)</a>	
<b>Sums two variables</b>	306
<a href="#">mqc_est::operator(+)</a>	314
<a href="#">mqc_est::operator(-)</a>	315
<a href="#">mqc_algebra::operator(-)</a>	
<b>Subtracts two variables</b>	315
<a href="#">mqc_algebra::operator(.dot.)</a>	
<b>Computes the inner product of two arrays</b>	323

<code>mqc_algebra::operator(.eq.)</code>	
<b>Determines if two variables are equal</b>	325
<code>mqc_algebra::operator(.ewd.)</code>	
<b>Computes the element-wise quotient of two arrays</b>	326
<code>mqc_algebra::operator(.ewp.)</code>	
<b>Computes the element-wise product of two arrays</b>	327
<code>mqc_algebra::operator(.ge.)</code>	
<b>Determines if a variable is greater than or equal to another</b>	329
<code>mqc_algebra::operator(.gt.)</code>	
<b>Determines if a variable is greater than another</b>	330
<code>mqc_algebra::operator(.le.)</code>	
<b>Determines if a variable is less than or equal to another</b>	335
<code>mqc_algebra::operator(.lt.)</code>	
<b>Determines if a variable is less than another</b>	340
<code>mqc_algebra::operator(.ne.)</code>	
<b>Determines if two variables are not equal</b>	342
<code>mqc_algebra::operator(.outer.)</code>	
<b>Computes the outer product of two vectors</b>	344
<code>mqc_algebra::operator(.x.)</code>	
<b>Computes the cross product of two vectors</b>	345
<code>mqc_algebra::operator(/)</code>	
<b>Divides two variables</b>	346
<code>mqc_algebra::real</code>	
<b>Returns the real part</b>	354
<code>mqc_algebra::sin</code>	
<b>Returns the sine</b>	356
<code>mqc_algebra::sqrt</code>	
<b>Returns the square root</b>	357
<code>mqc_algebra::tan</code>	
<b>Returns the tangent</b>	358
<code>mqc_est::transpose</code>	360
<code>mqc_algebra::transpose</code>	
<b>Returns the transpose</b>	360





## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

src/ <a href="#">mqc_algebra.F03</a> . . . . .	<a href="#">363</a>
src/ <a href="#">mqc_est.F03</a> . . . . .	<a href="#">376</a>



## Chapter 5

# Module Documentation

### 5.1 mqc\_algebra Module Reference

MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran

#### Data Types

- interface [abs](#)  
*Takes the absolute value*
- interface [acos](#)  
*Returns the arccosine*
- interface [aimag](#)  
*Returns the imaginary part*
- interface [asin](#)  
*Returns the arcsine*
- interface [assignment\(=\)](#)  
*Assigns a variable to the value of another*
- interface [atan](#)  
*Returns the arctangent*
- interface [atan2](#)  
*Returns the arctangent accounting for circle quadrant*
- interface [cmplx](#)  
*Defines a complex number*
- interface [conjg](#)  
*Returns the complex conjugate*
- interface [contraction](#)  
*Contracts two arrays*
- interface [cos](#)  
*Returns the cosine*
- interface [dagger](#)  
*Returns the Hermitian conjugate*

- interface [dot\\_product](#)  
*Returns the dot product*
- interface [matmul](#)  
*Multiplies two arrays*
- interface [matrix\\_symm2sq](#)  
*Sets a symmetric packed intrinsic array as a square packed intrinsic array*
- interface [mqc\\_cast\\_complex](#)  
*Sets an array to complex type*
- interface [mqc\\_cast\\_integer](#)  
*Sets an array to integer type*
- interface [mqc\\_cast\\_real](#)  
*Sets an array to real type*
- interface [mqc\\_have\\_complex](#)  
*Determines in an array is complex type*
- interface [mqc\\_have\\_int](#)  
*Determines in an array is integer type*
- interface [mqc\\_have\\_real](#)  
*Determines in an array is real type*
- type [mqc\\_matrix](#)  
*Rank 2 array variable*
- interface [mqc\\_matrix\\_diagmatrix\\_put](#)  
*Sets a diagonal packed intrinsic array as an MQC Matrix object*
- interface [mqc\\_matrix\\_symmmatrix\\_put](#)  
*Sets a symmetric packed intrinsic array as an MQC Matrix object*
- interface [mqc\\_print](#)  
*Prints an object*
- type [mqc\\_r4tensor](#)  
*Updates the specified element of the MQC Matrix to the specified value*
- type [mqc\\_scalar](#)  
*Rank 0 array variable*
- interface [mqc\\_set\\_array2vector](#)  
*Sets an intrinsic array as an MQC Algebra object*
- type [mqc\\_vector](#)  
*Rank 1 array variable*
- interface [operator\(\\*\)](#)  
*Multiplies two variables*
- interface [operator\(\\*\\*\)](#)  
*Exponentials a variable to the power of another*
- interface [operator\(+\)](#)  
*Sums two variables*
- interface [operator\(-\)](#)  
*Subtracts two variables*
- interface [operator\(.dot.\)](#)  
*Computes the inner product of two arrays*
- interface [operator\(.eq.\)](#)  
*Determines if two variables are equal*
- interface [operator\(.ewd.\)](#)

- *Computes the element-wise quotient of two arrays*
- interface [operator\(.ewp.\)](#)
- *Computes the element-wise product of two arrays*
- interface [operator\(.ge.\)](#)
- *Determines if a variable is greater than or equal to another*
- interface [operator\(.gt.\)](#)
- *Determines if a variable is greater than another*
- interface [operator\(.le.\)](#)
- *Determines if a variable is less than or equal to another*
- interface [operator\(.lt.\)](#)
- *Determines if a variable is less than another*
- interface [operator\(.ne.\)](#)
- *Determines if two variables are not equal*
- interface [operator\(.outer.\)](#)
- *Computes the outer product of two vectors*
- interface [operator\(.x.\)](#)
- *Computes the cross product of two vectors*
- interface [operator\(/\)](#)
- *Divides two variables*
- interface [real](#)
- *Returns the real part*
- interface [sin](#)
- *Returns the sine*
- interface [sqrt](#)
- *Returns the square root*
- interface [tan](#)
- *Returns the tangent*
- interface [transpose](#)
- *Returns the 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, Seed, Distribution)  
***MQC\_Scalar\_Get\_Random\_Value** is a function that returns a random real value from a specified distribution*
- 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, Intln)  
***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](#) (Intln, Scalar)  
***MQC\_IntegerGTScalar** is a function that returns TRUE if an intrinsic integer is greater than a MQC\_Scalar*
- logical function [mqc\\_scalargtinteger](#) (Scalar, Intln)  
***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](#) (Intln, 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, 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\\_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](#) (Realln, 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, Realln)
- 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)
- MQC\_Allocate\_Vector** is used to allocate a vector type variable of the MQC\_Vector class*

  - subroutine [mqc\\_deallocate\\_vector](#) (Vector)
- MQC\_Deallocate\_Vector** is used to deallocate a vector type variable of the MQC\_Vector class*

  - integer(kind=int64) function [mqc\\_length\\_vector](#) (Vector)
- MQC\_Length\_Vector** is used to return the length of an MQC vector*

  - logical function [mqc\\_vector\\_havereal](#) (Vector)
- MQC\_Vector\_HaveReal** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector*

  - logical function [mqc\\_vector\\_haveinteger](#) (Vector)
- MQC\_Vector\_HaveInteger** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector*

  - logical function [mqc\\_vector\\_havecomplex](#) (Vector)
- MQC\_Vector\_HaveComplex** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector*

  - logical function [mqc\\_vector\\_iscolumn](#) (Vector)
- MQC\_Vector\_IsColumn** is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector*

  - subroutine [mqc\\_vector\\_copy\\_int2real](#) (Vector)
- MQC\_Vector\_Copy\_Int2Real** is a subroutine that copies an integer MQC\_Vector into its real vector*

  - subroutine [mqc\\_vector\\_copy\\_int2complex](#) (Vector)
- MQC\_Vector\_Copy\_Int2Complex** is a subroutine that copies an integer MQC\_Vector into its complex vector*

  - subroutine [mqc\\_vector\\_copy\\_real2int](#) (Vector)
- MQC\_Vector\_Copy\_Real2Int** is a subroutine that copies a real MQC\_Vector into its integer vector*

  - subroutine [mqc\\_vector\\_copy\\_real2complex](#) (Vector)
- MQC\_Vector\_Copy\_Real2Complex** is a subroutine that copies a real MQC\_Vector into its complex vector*

  - subroutine [mqc\\_vector\\_copy\\_complex2int](#) (Vector)
- MQC\_Vector\_Copy\_Complex2Int** is a subroutine that copies a complex MQC\_Vector into its integer vector*

  - subroutine [mqc\\_vector\\_copy\\_complex2real](#) (Vector)
- MQC\_Vector\_Copy\_Complex2Real** is a subroutine that copies a complex MQC\_Vector into its real vector*

  - type([mqc\\_scalar](#)) function [mqc\\_vector\\_scalar\\_at](#) (Vec, I)
- MQC\_Vector\_Scalar\_At** is a function that returns the ith element of a MQC vector as an MQC scalar*

  - type([mqc\\_vector](#)) function [mqc\\_vector\\_vector\\_at](#) (Vec, I, J)
- MQC\_Vector\_Vector\_At** is a function that returns the vector at the specified subvector of MQC\_Vector*

  - subroutine [mqc\\_set\\_vector2integerarray](#) (ArrayOut, VectorIn)
- MQC\_Set\_Vector2IntegerArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array*

  - subroutine [mqc\\_set\\_vector2realarray](#) (ArrayOut, VectorIn)

***MQC\_Set\_Vector2RealArray* is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array**

- subroutine [mqc\\_set\\_vector2complexarray](#) (ArrayOut, VectorIn)

***MQC\_Set\_Vector2ComplexArray* is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array**

- subroutine [mqc\\_set\\_array2vector\\_integer](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Integer* is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector**

- subroutine [mqc\\_set\\_array2vector\\_real](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Real* is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector**

- subroutine [mqc\\_set\\_array2vector\\_complex](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Complex* is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector**

- subroutine [mqc\\_set\\_vector2vector](#) (VectorOut, VectorIn)

***MQC\_Set\_Vector2Vector* is a subroutine that sets a MQC vector equal to another MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_vectorvectorsum](#) (Vector1In, Vector2In)

***MQC\_VectorVectorSum* is a function that adds two MQC vectors and stores them in another MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_vectorvectordifference](#) (Vector1In, Vector2In)

***MQC\_VectorVectorDifference* is a function that subtracts two MQC vectors and stores them in another MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_scalarvectorsum](#) (ScalarIn, VectorIn)

***MQC\_ScalarVectorSum* is a function that adds an MQC scalar to all elements of an MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_scalarvectordifference](#) (ScalarIn, VectorIn)

***MQC\_ScalarVectorDifference* is a function that subtracts an MQC scalar from all elements of an MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_elementvectorproduct](#) (Vector1In, Vector2In)

***MQC\_ElementVectorProduct* is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_vector\\_transpose](#) (Vector)

***MQC\_Vector\_Transpose* is a function that returns the transpose of an MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_vector\\_conjugate\\_transpose](#) (Vector)

***MQC\_Vector\_Conjugate\_Transpose* is a function that returns the conjugate transpose of an MQC vector**

- type([mqc\\_scalar](#)) function [mqc\\_vectorvectordotproduct](#) (Vector1, Vector2)

***MQC\_VectorVectorDotProduct* is a function that returns the dot product of two MQC vectors**

- type([mqc\\_matrix](#)) function [mqc\\_outer](#) (VA, VB)

***MQC\_Outer* is a function that returns the outer product of two MQC vectors**

- type([mqc\\_vector](#)) function [mqc\\_crossproduct](#) (Vector1In, Vector2In)

***MQC\_CrossProduct* is a function that returns the cross product of two MQC vectors**

- subroutine [mqc\\_print\\_vector\\_algebra1](#) (Vector, IOut, Header, Verbose, Blank\_At\_Top, Blank\_At\_Bottom)

***MQC\_Print\_Vector\_Algebra1* is a subroutine used to print an MQC vector**

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_integer](#) (VA)

***MQC\_vector\_cast\_integer* is a function that converts an MQC vector to its integer space**

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_real](#) (VA)

***MQC\_vector\_cast\_real* is a function that converts an MQC vector to its real space**

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_complex](#) (VA)

***MQC\_vector\_cast\_complex* is a function that converts an MQC vector to its complex space**

- subroutine [mqc\\_vector\\_scalar\\_put](#) (Vector, Scalar, I)

***MQC\_Vector\_Scalar\_Put* is a subroutine that updates the value of the ith element of a MQC vector with the value of a MQC scalar**

- subroutine [mqc\\_vector\\_scalar\\_increment](#) (Vector, Scalar, I)

***MQC\_Vector\_Scalar\_Increment* is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar**

- subroutine [mqc\\_vector\\_vector\\_put](#) (Vector, VectorIn, I)

*MQC\_Vector\_Vector\_Put* is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector

- subroutine `mqc_vector_initialize` (Vector, Length, Scalar)

*MQC\_Vector\_Initialize* is a subroutine that initializes a MQC vector

- type(`mqc_vector`) function `mqc_scalarvectorproduct` (Scalar, Vector)

*MQC\_ScalarVectorProduct* is a function that returns the product of a MQC scalar with a MQC vector

- type(`mqc_vector`) function `mqc_vectorscalarproduct` (vector, scalar)

*MQC\_VectorScalarProduct* is a function that returns the product of a MQC vector with a MQC scalar

- type(`mqc_vector`) function `mqc_vectorscalardivide` (vector, scalar)

*MQC\_VectorScalarDivide* is a function that returns a MQC vector divided by a MQC scalar

- type(`mqc_vector`) function `mqc_realvectorproduct` (Realln, Vector)

*MQC\_RealVectorProduct* is a function that returns the product of an intrinsic real scalar and a MQC vector

- type(`mqc_vector`) function `mqc_vectorrealproduct` (vector, realln)

*MQC\_VectorRealProduct* is a function that returns the product of a MQC vector and an intrinsic real scalar

- type(`mqc_vector`) function `mqc_vectorrealdivide` (vector, realln)

*MQC\_VectorRealDivide* is a function that returns a MQC vector divided by an intrinsic real integer

- type(`mqc_vector`) function `mqc_integervectorproduct` (intln, Vector)

*MQC\_IntegerVectorProduct* is a function that returns the product of an intrinsic integer scalar and a MQC vector

- type(`mqc_vector`) function `mqc_vectorintegerproduct` (vector, intln)

*MQC\_VectorIntegerProduct* is a function that returns the product of a MQC vector and an intrinsic integer scalar

- type(`mqc_vector`) function `mqc_vectorintegerdivide` (vector, intln)

*MQC\_VectorIntegerDivide* is a function that returns a MQC vector divided by an intrinsic integer scalar

- type(`mqc_vector`) function `mqc_complexvectorproduct` (Compln, Vector)

*MQC\_ComplexVectorProduct* is a function that returns the product of an intrinsic complex scalar and a MQC vector

- type(`mqc_vector`) function `mqc_vectorcomplexproduct` (vector, compln)

*MQC\_VectorComplexProduct* is a function that returns the product of a MQC vector and an intrinsic complex scalar

- type(`mqc_vector`) function `mqc_vectorcomplexdivide` (vector, compln)

*MQC\_VectorComplexDivide* is a function that returns a MQC vector divided by an intrinsic complex scalar

- type(`mqc_scalar`) function `mqc_vector_norm` (vector, methodln)

*MQC\_Vector\_Norm* is a function that returns the norm of a MQC vector

- logical function `mqc_vector_isallocated` (Vector)

*MQC\_Vector\_isAllocated* is a function that returns TRUE if a MQC vector is allocated and FALSE if it is not

- subroutine `mqc_vector_push` (Vector, Scalar)

*MQC\_Vector\_Push* is a function that adds a value to the end of a MQC vector

- subroutine `mqc_vector_unshift` (Vector, Scalar)

*MQC\_Vector\_Unshift* is a function that adds a value to the beginning of a MQC vector

- type(`mqc_scalar`) function `mqc_vector_pop` (Vector)

*MQC\_Vector\_Pop* is a function that removes a value from the end of a MQC vector and returns it

- type(`mqc_scalar`) function `mqc_vector_shift` (Vector)

*MQC\_Vector\_Shift* is a function that removes a value from the beginning of a MQC vector and returns it

- type(`mqc_scalar`) function `mqc_vector_maxval` (Vector)

*MQC\_Vector\_MaxVal* is a function that returns the largest value in a MQC vector

- type([mqc\\_scalar](#)) function [mqc\\_vector\\_minval](#) (Vector)  
***MQC\_Vector\_MinVal** is a function that returns the smallest value in an MQC vector*
- integer function [mqc\\_vector\\_maxloc](#) (Vector)  
***MQC\_Vector\_MaxLoc** is a function that returns the index of the largest value in an MQC vector*
- integer function [mqc\\_vector\\_minloc](#) (Vector)  
***MQC\_Vector\_MinLoc** is a function that returns the index of the smallest value in an MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vector\\_argsort](#) (Vector)  
***MQC\_Vector\_Argsort** is a function that returns the indices of an MQC vector sorted from low to high*
- subroutine [mqc\\_vector\\_sort](#) (Vector, idx)  
***MQC\_Vector\_Sort** is a function that returns an MQC vector sorted from low to high unless optional index order is present*
- subroutine [mqc\\_vector\\_sqrt](#) (A)  
***MQC\_Vector\_Sqrt** is a function that returns the square root of all elements of an MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vector\\_abs](#) (A)  
***MQC\_Vector\_Abs** is a function that returns the absolute value of all elements of an MQC vector*
- subroutine [mqc\\_vector\\_power](#) (A, P)  
***MQC\_Vector\_Power** is a function that returns the value of all elements of an MQC vector raised to a power*
- type([mqc\\_vector](#)) function [mqc\\_vector\\_complex\\_realpart](#) (A)  
***MQC\_Vector\_Complex\_RealPart** is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vector\\_complex\\_imagpart](#) (A)  
***MQC\_Vector\_Complex\_ImagPart** is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vector\\_cmplx](#) (Vector1, Vector2)  
***MQC\_Vector\_Cmplx** is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector*
- subroutine [mqc\\_matrix\\_diagonalize](#) (A, EVals, EVecs)  
***MQC\_Matrix\_Diagonalize** is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_real](#) (MA)  
***MQC\_Matrix\_Cast\_Real** is a function that converts an MQC matrix to its real space*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_integer](#) (MA)  
***MQC\_Matrix\_Cast\_Integer** is a function that converts an MQC matrix to its integer space*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_complex](#) (MA)  
***MQC\_Matrix\_Cast\_Complex** is a function that converts an MQC matrix to its complex space*
- type([mqc\\_scalar](#)) function [mqc\\_matrix\\_scalar\\_at](#) (Mat, I, J)  
***MQC\_Matrix\_Scalar\_At** is a function that returns the value of an element of a MQC matrix*
- type([mqc\\_vector](#)) function [mqc\\_matrix\\_vector\\_at](#) (Mat, Rows, Cols)  
***MQC\_Matrix\_Vector\_At** is a function that returns the subvector of an MQC matrix*
- recursive subroutine [mqc\\_matrix\\_vector\\_put](#) (Mat, VectorIn, Rows, Cols)  
***MQC\_Matrix\_Vector\_Put** is a subroutine that writes a subvector to the specified position of a MQC matrix*
- 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)  
***MQC\_Matrix\_DiagMatrix\_Put\_Vector** is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector*

- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_integer](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_integer** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_real](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_Real** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_complex](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_Complex** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector*
- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_integer](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Integer** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_real](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Real** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_complex](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Complex** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector*
- recursive subroutine [mqc\\_matrix\\_matrix\\_put](#) (Mat, MatrixIn, Rows, Cols)  
***MQC\_Matrix\_Matrix\_Put** is a subroutine that writes a submatrix to the specified position of a MQC matrix*
- integer(kind=int64) function [symindexhash](#) (i, j, k, l)  
***SymIndexHash** is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor*
- type([mqc\\_matrix](#)) function [mqc\\_elementmatrixproduct](#) (A, B)  
***MQC\_ElementMatrixProduct** is a function that returns the element- wise product of two MQC matrices*
- type([mqc\\_matrix](#)) function [mqc\\_elementmatrixdivide](#) (A, B)  
***MQC\_ElementMatrixDivide** is a function that returns the element- wise quotient of two MQC matrices*
- logical function [mqc\\_matrix\\_test\\_symmetric](#) (Matrix, Option)  
***MQC\_Matrix\_Test\_Symmetric** is a function that tests a MQC matrix for symmetry*
- logical function [mqc\\_matrix\\_test\\_diagonal](#) (Matrix)  
***MQC\_Matrix\_Test\_Diagonal** is a function that tests a MQC matrix to determine if it is diagonal*
- subroutine [mqc\\_allocate\\_matrix](#) (M, N, Matrix, Data\_Type, Storage)  
***MQC\_Allocate\_Matrix** is used to allocate a matrix type variable of the MQC\_Matrix class*
- subroutine [mqc\\_deallocate\\_matrix](#) (Matrix)  
***MQC\_Deallocate\_Matrix** is used to deallocate a matrix type variable of the MQC\_Matrix class*
- logical function [mqc\\_matrix\\_isallocated](#) (Matrix)  
***MQC\_Matrix\_isAllocate** is a function that returns the allocation status of a MQC\_Matrix variable*
- subroutine [mqc\\_set\\_integerarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_IntegerArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array*
- subroutine [mqc\\_set\\_realarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_RealArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array*
- subroutine [mqc\\_set\\_complexarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_ComplexArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array*
- subroutine [mqc\\_set\\_matrix2integerarray](#) (ArrayOut, MatrixIn)  
***MQC\_Set\_Matrix2IntegerArray** is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix*
- subroutine [mqc\\_set\\_matrix2realarray](#) (ArrayOut, MatrixIn)  
***MQC\_Set\_Matrix2RealArray** is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix*
- subroutine [mqc\\_set\\_matrix2complexarray](#) (ArrayOut, MatrixIn)

***MQC\_Set\_Matrix2ComplexArray* is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix**

- subroutine [mqc\\_set\\_matrix2matrix](#) (MatrixOut, MatrixIn)

***MQC\_Set\_Matrix2Matrix* is a subroutine that sets an MQC matrix equal to another MQC matrix**

- subroutine [mqc\\_print\\_matrix\\_algebra1](#) (Matrix, IOut, Header, Blank\_At\_Top, Blank\_At\_Bottom)

***MQC\_Print\_Matrix\_Algebra1* is a subroutine used to print an MQC matrix**

- subroutine [mqc\\_matrix\\_copy\\_int2real](#) (Matrix)

***MQC\_Matrix\_Copy\_Int2Real* is a subroutine used to copy an integer MQC matrix into its real space**

- subroutine [mqc\\_matrix\\_copy\\_int2complex](#) (Matrix)

***MQC\_Matrix\_Copy\_Int2Complex* is a subroutine used to copy an integer MQC matrix into its complex space**

- subroutine [mqc\\_matrix\\_copy\\_real2int](#) (Matrix)

***MQC\_Matrix\_Copy\_Real2Int* is a subroutine used to copy a real MQC matrix into its integer space**

- subroutine [mqc\\_matrix\\_copy\\_real2complex](#) (Matrix)

***MQC\_Matrix\_Copy\_Real2Complex* is a subroutine used to copy a real MQC matrix into its complex space**

- subroutine [mqc\\_matrix\\_copy\\_complex2int](#) (Matrix)

***MQC\_Matrix\_Copy\_Complex2Int* is a subroutine used to copy a complex MQC matrix into its integer space**

- subroutine [mqc\\_matrix\\_copy\\_complex2real](#) (Matrix)

***MQC\_Matrix\_Copy\_Complex2Real* is a subroutine used to copy a complex MQC matrix into its real space**

- integer(kind=int64) function [mqc\\_matrix\\_rows](#) (Matrix)

***MQC\_Matrix\_Rows* is a function used to return the number of rows of an MQC matrix**

- integer(kind=int64) function [mqc\\_matrix\\_columns](#) (Matrix)

***MQC\_Matrix\_Columns* is a function used to return the number of columns of an MQC matrix**

- logical function [mqc\\_matrix\\_havereal](#) (Matrix)

***MQC\_Matrix\_HaveReal* is a function used to indicate if an MQC matrix has an allocated real matrix**

- logical function [mqc\\_matrix\\_haveinteger](#) (Matrix)

***MQC\_Matrix\_HaveInteger* is a function used to indicate if an MQC matrix has an allocated integer matrix**

- logical function [mqc\\_matrix\\_havecomplex](#) (Matrix)

***MQC\_Matrix\_HaveComplex* is a function used to indicate if an MQC matrix has an allocated complex matrix**

- logical function [mqc\\_matrix\\_havfull](#) (Matrix)

***MQC\_Matrix\_HaveFull* is a function used to indicate if an MQC matrix is stored unpacked**

- logical function [mqc\\_matrix\\_havesymmetric](#) (Matrix)

***MQC\_Matrix\_HaveSymmetric* is a function used to indicate if an MQC matrix is stored symmetric-packed**

- logical function [mqc\\_matrix\\_havediagonal](#) (Matrix)

***MQC\_Matrix\_HaveDiagonal* is a function used to indicate if an MQC matrix is stored diagonal-packed**

- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_transpose](#) (Matrix)

***MQC\_Matrix\_Transpose* is a function that returns the transpose of a MQC matrix**

- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_conjugate\\_transpose](#) (Matrix)

***MQC\_Matrix\_Conjugate\_Transpose* is a function that returns the conjugate transpose of a MQC matrix**

- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_symmetrize](#) (Matrix)

***MQC\_Matrix\_Symmetrize* is a function that symmetrizes a MQC matrix**

- subroutine [mqc\\_matrix\\_full2symm](#) (Matrix)

***MQC\_Matrix\_Full2Symm* is a subroutine that converts an unpacked MQC matrix to symmetric-packed**

- subroutine [mqc\\_matrix\\_symm2full](#) (Matrix, Option)

***MQC\_Matrix\_Symm2Full* is a subroutine that converts a symmetry-packed MQC matrix to unpacked**

- subroutine [mqc\\_matrix\\_full2diag](#) (Matrix)

***MQC\_Matrix\_Full2Diag* is a subroutine that converts an unpacked MQC matrix to diagonal-packed**

- subroutine [mqc\\_matrix\\_diag2full](#) (Matrix)



***MQC\_Matrix\_Diag2Full* is a subroutine that converts a diagonal-packed MQC matrix to unpacked**

- subroutine [mqc\\_matrix\\_symm2diag](#) (Matrix)

***MQC\_Matrix\_Symm2Diag* is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed**

- subroutine [mqc\\_matrix\\_diag2symm](#) (Matrix)

***MQC\_Matrix\_Diag2Symm* is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed**

- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_symm2full\\_func](#) (Matrix)

***MQC\_Matrix\_Symm2Full\_Func* is a function that converts a symmetric- packed MQC matrix to unpacked**

- subroutine [matrix\\_symm2sq\\_integer](#) (N, I\_Symm, I\_Sq)

***Matrix\_Symm2Sq\_Integer* is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array**

- subroutine [matrix\\_symm2sq\\_real](#) (N, A\_Symm, A\_Sq)

***Matrix\_Symm2Sq\_Real* is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array**

- subroutine [matrix\\_symm2sq\\_complex](#) (N, A\_Symm, A\_Sq)

***Matrix\_Symm2Sq\_Complex* is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array**

- type([mqc\\_matrix](#)) function [mqc\\_vector2diagmatrix](#) (vector)

***MQC\_Vector2DiagMatrix* is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector**

- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixsum](#) (MA, MB)

***MQC\_MatrixMatrixSum* is a function that sums two MQC matrices**

- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixsubtract](#) (MA, MB)

***MQC\_MatrixMatrixSubtract* is a function that subtracts two MQC matrices**

- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixproduct](#) (MA, MB)

***MQC\_MatrixMatrixProduct* is a function that computes the element- wise product of two MQC matrices**

- 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 Detailed Description

**MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran**

**Purpose:**

MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran. Arrays can be packed for efficient memory use and used for operations completely transparently to the user. Furthermore, there is no need to type arrays, as this can be manipulated on the fly. Arrays carry their own procedures, and use underlying lapack routines for efficiency. The MQC derived types defined in this package are:

1. MQC\_Scalar: Rank 0 array variable
2. MQC\_Vector: Rank 1 array variable
3. MQC\_Matrix: Rank 2 array variable
4. MQC\_R4Tensor: Rank 3 array variable

This module is level 1 in the MQC hierarchy and so depends on level 0 modules.

Note that MQC\_Algebra2 provides similar functionality with MQC\_Array objects that can dynamically adjust rank. However, the vast majority of work can be performed using MQC\_Algebra derived types which have been more developed than MQC\_Algebra2 derived types.

### 5.1.2 Function/Subroutine Documentation

#### 5.1.2.1 `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	$N$	<code>N is Integer(kind=int64)</code> The number of objects.
in	$K$	<code>K is Integer(kind=int64)</code> The number of permutations.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.2 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	$N$	<code>N is Integer(kind=int64)</code> The argument of the factorial function.
----	-----	--

**Author**

L. M. Thompson

**Date**

2016

### 5.1.2.3 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 )
```

**Matrix\_Symm2Sq\_Complex** is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array

#### Purpose:

Matrix\_Symm2Sq\_Complex is a subroutine that converts a symmetric-packed intrinsic complex matrix to a rank-2 complex array.  
 TODO: Move this routine to MQC general

#### Parameters

in	<i>N</i>	N is Integer(kind=int64) The leading dimension of symmetric-packed matrix I_Symm. unpacked.
in	<i>A_Symm</i>	A_Symm is Complex(kind=real64),Dimension(:) The symmetric-packed intrinsic complex matrix to be unpacked.
out	<i>A_Sq</i>	A_Sq is Complex(kind=real64),Dimension(N,N) The unpacked intrinsic complex matrix output.

#### Author

L. M. Thompson

#### Date

2017

### 5.1.2.4 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 )
```

**Matrix\_Symm2Sq\_Integer** is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array

**Purpose:**

Matrix\_Symm2Sq\_Integer is a subroutine that converts a symmetric-packed intrinsic integer matrix to a rank-2 integer array.  
 TODO: Move this routine to MQC general

**Parameters**

in	<i>N</i>	N is Integer(kind=int64) The leading dimension of symmetric-packed matrix I_Symm. unpacked.
in	<i>I_Symm</i>	I_Symm is Integer(kind=int64),Dimension(:) The symmetric-packed intrinsic integer matrix to be unpacked.
out	<i>I_Sq</i>	I_Sq is Integer(kind=int64),Dimension(N,N) The unpacked intrinsic integer matrix output.

**Author**

H. P. Hratchian

**Date**

2017

**5.1.2.5 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 )
```

**Matrix\_Symm2Sq\_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array**

**Purpose:**

Matrix\_Symm2Sq\_Real is a subroutine that converts a symmetric-packed intrinsic real matrix to a rank-2 real array.  
 TODO: Move this routine to MQC general

**Parameters**

in	<i>N</i>	N is Integer(kind=int64) The leading dimension of symmetric-packed matrix I_Symm. unpacked.
in	<i>A_Symm</i>	A_Symm is Real(kind=real64),Dimension(:) The symmetric-packed intrinsic real matrix to be unpacked.
out	<i>A_Sq</i>	A_Sq is Real(kind=real64),Dimension(N,N) The unpacked intrinsic real matrix output.

**Author**

H. P. Hratchian

**Date**

2017

**5.1.2.6 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 )

```

**MQC\_Allocate\_Matrix** is used to allocate a matrix type variable of the **MQC\_Matrix** class

**Purpose:**

MQC\_Allocate\_Matrix is a subroutine used to allocate a matrix type variable of the MQC\_Matrix class. The following options are available:

1. Data\_Type = 'Real' declares the MQC\_Matrix variable to be of real type.
2. Data\_Type = 'Integer' declares the MQC\_Matrix variable to be of integer type.
3. Data\_Type = 'Complex' declares the MQC\_Matrix variable to be of complex type.
1. Data\_Type = 'StorFull' declares the MQC\_Matrix variable to be unpacked.
2. Data\_Type = 'StorSymm' declares the MQC\_Matrix variable to be symmetric packed.
3. Data\_Type = 'StorDiag' declares the MQC\_Matrix variable to be diagonal packed.

## Parameters

in	<i>M</i>	M is Integer(kind=int64) M is the number of rows of Matrix.
in	<i>N</i>	N is Integer(kind=int64) N is the number of columns of Matrix.
in, out	<i>Matrix</i>	Matrix is Class(MQC_Matrix) The MQC matrix to be allocated.
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.
in	<i>Storage</i>	Storage is Character(Len=*) = 'StorFull': the MQC_Scalar is unpacked = 'StorSymm': the MQC_Scalar is symmetric packed = 'StorDiag': the MQC_Scalar is diagonal packed.

## Author

H. P. Hratchian

L. M. Thompson

## Date

2016

## 5.1.2.7 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 )

```

### 5.1.2.8 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

### 5.1.2.9 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 )
```

**MQC\_Allocate\_Vector** is used to allocate a vector type variable of the **MQC\_Vector** class

**Purpose:**

MQC\_Allocate\_Vector is a subroutine used to allocate a vector type variable of the MQC\_Vector class. The following options are available:

1. Data\_Type = 'Real' declares the MQC\_Vector variable to be of real type.
2. Data\_Type = 'Integer' declares the MQC\_Vector variable to be of integer type.
3. Data\_Type = 'Complex' declares the MQC\_Vector variable to be of complex type.

**Parameters**

in	<i>N</i>	N is Integer(kind=int64) The length of the MQC_Vector variable
in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The name of the MQC_Vector variable
in	<i>Data_Type</i>	Data_Type is Character(Len=*) = 'Real': the MQC_Vector is real = 'Integer': the MQC_Vector is integer = 'Complex': the MQC_Vector is complex

**Author**

H. P. Hratchian

**Date**

2016

**5.1.2.10 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>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

**5.1.2.11 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↔ 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

### 5.1.2.12 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↔ 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

### 5.1.2.13 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>Complex↔ 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

**5.1.2.14 mqc\_complexvectorproduct()**

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

**MQC\_ComplexVectorProduct** is a function that returns the product of an intrinsic complex scalar and a MQC vector

**Purpose:**

MQC\_ComplexVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector.

**Parameters**

in	<i>Comp↔ In</i>	CompIn is Complex(kind=real64) The intrinsic complex to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.15 mqc\_crossproduct()**

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

**MQC\_CrossProduct** is a function that returns the cross product of two MQC vectors

**Purpose:**

MQC\_CrossProduct is a function that returns the cross product of two MQC vectors. The vectors should both be of length 3.

**Parameters**

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector.

**Author**

L. M. Thompson

**Date**

2016

### 5.1.2.16 mqc\_deallocate\_matrix()

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

**MQC\_Deallocate\_Matrix** is used to deallocate a matrix type variable of the **MQC\_Matrix** class

#### Purpose:

MQC\_Deallocate\_Matrix is a subroutine used to deallocate a matrix type variable of the MQC\_Matrix class.

#### Parameters

in, out	<i>Matrix</i>	Matrix is Class(MQC_Matrix) The MQC matrix to be deallocated.
---------	---------------	--

#### Author

L. M. Thompson

#### Date

2016

### 5.1.2.17 mqc\_deallocate\_r4tensor()

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

### 5.1.2.18 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	<i>Scalar</i>	<p>Scalar is Type(MQC_Scalar) The name of the MQC_Scalar variable to deallocate.</p>
---------	---------------	--

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.19 mqc\_deallocate\_vector()**

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

**MQC\_Deallocate\_Vector** is used to deallocate a vector type variable of the MQC\_Vector class

**Purpose:**

MQC\_Deallocate\_Vector is a subroutine used to deallocate a vector type variable of the MQC\_Vector class.

**Parameters**

in, out	<i>Vector</i>	<p>Vector is Type(MQC_Vector) The name of the MQC_Vector variable to deallocate.</p>
---------	---------------	--

**Author**

H. P. Hratchian

**Date**

2016

### 5.1.2.20 mqc\_elementmatrixdivide()

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

**MQC\_ElementMatrixDivide** is a function that returns the element- wise quotient of two MQC matrices

#### Purpose:

MQC\_ElementMatrixDivide is a function that returns the element-wise quotient of two MQC matrices.

#### Parameters

in	<i>A</i>	A is type(mqc_matrix) The matrix with elements being the numerator.
in	<i>B</i>	B is type(mqc_matrix) The matrix with elements being the denominator.

#### Author

X. Sheng

#### Date

2017

### 5.1.2.21 mqc\_elementmatrixproduct()

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

**MQC\_ElementMatrixProduct** is a function that returns the element- wise product of two MQC matrices

#### Purpose:

MQC\_ElementMatrixProduct is a function that returns the element-wise product of two MQC matrices.

## Parameters

in	<i>A</i>	A is type(mqc_matrix) The first matrix to element-wise multiply.
in	<i>B</i>	B is type(mqc_matrix) The second matrix to element-wise multiply.

## Author

X. Sheng

## Date

2017

## 5.1.2.22 mqc\_elementvectorproduct()

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

**MQC\_ElementVectorProduct** is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector

## Purpose:

MQC\_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector.

## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The frist MQC vector to multiply elementwise.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector to multiply elementwise.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.23 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 )

```

**5.1.2.24 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**

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

**Author**

L. M. Thompson



Date

2017

#### 5.1.2.25 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

#### 5.1.2.26 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

**5.1.2.27 mqc\_integergtscalar()**

```
logical function mqc_algebra::mqc_integergtscalar (
    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

**5.1.2.28 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

### 5.1.2.29 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>IntegerIn</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

### 5.1.2.30 mqc\_integerscalardivide()

```
type(mqc_scalar) function mqc_algebra::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>Integer↔ In</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

## 5.1.2.31 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

### 5.1.2.32 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>IntegerIn</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

### 5.1.2.33 mqc\_integervectorproduct()

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

**MQC\_IntegerVectorProduct** is a function that returns the product of an intrinsic integer scalar and a MQC vector

Purpose:

MQC\_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector.

**Parameters**

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.34 mqc\_length\_vector()**

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

**MQC\_Length\_Vector is used to return the length of an MQC vector****Purpose:**

MQC\_Length\_Vector is used to return the length of an MQC vector. If the vector vector is NOT allocated, the length is returned as 0.

**Parameters**

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The name of the MQC_Vector variable whose length will be returned.
---------	---------------	--

**Author**

H. P. Hratchian

**Date**

2016

### 5.1.2.35 mqc\_matrix\_cast\_complex()

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

**MQC\_Matrix\_Cast\_Complex** is a function that converts an MQC matrix to its complex space

#### Purpose:

MQC\_Matrix\_Cast\_Complex is a function that converts an MQC matrix to its complex space.

#### Parameters

in	<i>MA</i>	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	-----------	--

#### Author

L. M. Thompson

#### Date

2017

### 5.1.2.36 mqc\_matrix\_cast\_integer()

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

**MQC\_Matrix\_Cast\_Integer** is a function that converts an MQC matrix to its integer space

#### Purpose:

MQC\_Matrix\_Cast\_Integer is a function that converts an MQC matrix to its integer space.

#### Parameters

in	<i>MA</i>	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	-----------	--



**Author**

L. M. Thompson

**Date**

2019

**5.1.2.37 mqc\_matrix\_cast\_real()**

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

**MQC\_Matrix\_Cast\_Real** is a function that converts an MQC matrix to its real space

**Purpose:**

MQC\_Matrix\_Cast\_Real is a function that converts an MQC matrix to its real space.

**Parameters**

in	MA	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	----	--

**Author**

X. Sheng

**Date**

2017

**5.1.2.38 mqc\_matrix\_columns()**

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

**MQC\_Matrix\_Columns** is a function used to return the number of columns of an MQC matrix

**Purpose:**

MQC\_Matrix\_Columns is a function used to return the number of columns of an MQC matrix. If the matrix is NOT allocated, the number of columns is returned as 0.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.39 mqc\_matrix\_conjugate\_transpose()**

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

**MQC\_Matrix\_Conjugate\_Transpose** is a function that returns the conjugate transpose of a MQC matrix

**Purpose:**

MQC\_Matrix\_Conjugate\_Transpose is a function that returns the conjugate transpose of a MQC matrix.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be conjugate transposed.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

#### 5.1.2.40 mqc\_matrix\_copy\_complex2int()

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

**MQC\_Matrix\_Copy\_Complex2Int** is a subroutine used to copy a complex MQC matrix into its integer space

**Purpose:**

MQC\_Matrix\_Copy\_Complex2Int is a subroutine used to copy a complex MQC matrix into its integer space.

**Parameters**

in	<i>Matrix</i>	
		Matrix is Type(MQC_Matrix) The MQC matrix to be converted to integer.

**Author**

L. M. Thompson

**Date**

2017

#### 5.1.2.41 mqc\_matrix\_copy\_complex2real()

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

**MQC\_Matrix\_Copy\_Complex2Real** is a subroutine used to copy a complex MQC matrix into its real space

**Purpose:**

MQC\_Matrix\_Copy\_Complex2Real is a subroutine used to copy a complex MQC matrix into its real space.

**Parameters**

in	<i>Matrix</i>	
		Matrix is Type(MQC_Matrix) The MQC matrix to be converted to real.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.42 mqc\_matrix\_copy\_int2complex()**

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

**MQC\_Matrix\_Copy\_Int2Complex** is a subroutine used to copy an integer MQC matrix into its complex space

**Purpose:**

MQC\_Matrix\_Copy\_Int2Complex is a subroutine used to copy an integer MQC matrix into its complex space.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to complex.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.43 mqc\_matrix\_copy\_int2real()**

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

**MQC\_Matrix\_Copy\_Int2Real** is a subroutine used to copy an integer MQC matrix into its real space

**Purpose:**

MQC\_Matrix\_Copy\_Int2Real is a subroutine used to copy an integer MQC matrix into its real space.

## Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to real.
----	---------------	---

## Author

L. M. Thompson

## Date

2016

**5.1.2.44 mqc\_matrix\_copy\_real2complex()**

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

**MQC\_Matrix\_Copy\_Real2Complex** is a subroutine used to copy a real MQC matrix into its complex space

## Purpose:

MQC\_Matrix\_Copy\_Real2Complex is a subroutine used to copy a real MQC matrix matrix into its complex space.

## Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to complex.
----	---------------	--

## Author

L. M. Thompson

## Date

2017

#### 5.1.2.45 mqc\_matrix\_copy\_real2int()

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

**MQC\_Matrix\_Copy\_Real2Int** is a subroutine used to copy a real MQC matrix into its integer space

##### Purpose:

MQC\_Matrix\_Copy\_Real2Int is a subroutine used to copy a real MQC matrix matrix into its integer space.

##### Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to integer.
----	---------------	--

##### Author

L. M. Thompson

##### Date

2016

#### 5.1.2.46 mqc\_matrix\_determinant()

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

#### 5.1.2.47 mqc\_matrix\_diag2full()

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

**MQC\_Matrix\_Diag2Full** is a subroutine that converts a diagonal-packed MQC matrix to unpacked

##### Purpose:

MQC\_Matrix\_Diag2Full is a subroutine that converts a diagonal-packed MQC matrix to unpacked.  
TODO: make tests for diagonal structure more efficient.

## Parameters

in, out	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.
---------	---------------	--

## Author

L. M. Thompson

## Date

2017

## 5.1.2.48 mqc\_matrix\_diag2symm()

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

**MQC\_Matrix\_Diag2Symm** is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed

## Purpose:

MQC\_Matrix\_Diag2Symm is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed.  
TODO: make tests for diagonal structure more efficient.

## Parameters

in, out	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.
---------	---------------	--

## Author

L. M. Thompson

## Date

2017

#### 5.1.2.49 `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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_Complex** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector

##### Purpose:

MQC\_Matrix\_DiagMatrix\_Put\_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector.

##### Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>DiagMatrixIn</i>	DiagMatrixIn is complex(kind=real64),dimension(:) Intrinsic complex vector to write as diagonal matrix.

##### Author

L. M. Thompson

##### Date

2017

#### 5.1.2.50 `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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_integer** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector

##### Purpose:

MQC\_Matrix\_DiagMatrix\_Put\_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector.



## Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>Diag↔ MatrixIn</i>	DiagMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as diagonal matrix.

## Author

H. P. Hratchian

## Date

2017

## 5.1.2.51 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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_Real** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector

## Purpose:

MQC\_Matrix\_DiagMatrix\_Put\_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector.

## Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>Diag↔ MatrixIn</i>	DiagMatrixIn is real(kind=real64),dimension(:) Intrinsic real vector to write as diagonal matrix.

**Author**

H. P. Hratchian

**Date**

2017

**5.1.2.52 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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_Vector** is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

**Purpose:**

MQC\_Matrix\_DiagMatrix\_Put\_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector.

**Parameters**

in	<i>Diag↔ VectorIn</i>	DiagVectorIn is class(MQC_Vector) Name of the MQC vector to write as diagonal matrix.
in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.

**Author**

L. M. Thompson

**Date**

2018

**5.1.2.53 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 )
```

**MQC\_Matrix\_Diagonalize** is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors

**Purpose:**

MQC\_Matrix\_Diagonalize is a subroutine that takes a symmetric or hermitian MQC matrix and optionally returns eigenvalues to a MQC vector and/or eigenvectors to a MQC matrix.

**Parameters**

in	<i>A</i>	A is Class(MQC_Matrix) The MQC matrix to diagonalize.
in, out	<i>EVals</i>	EVals is Type(MQC_Vector), Optional Optional MQC vector containing the eigenvalues.
in, out	<i>EVecs</i>	EVecs is Type(MQC_Vector), Optional Optional MQC matrix containing the eigenvectors.

**Author**

X. Sheng  
L. M. Thompson

**Date**

2017

**5.1.2.54 mqc\_matrix\_full2diag()**

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

**MQC\_Matrix\_Full2Diag** is a subroutine that converts an unpacked MQC matrix to diagonal-packed

**Purpose:**

MQC\_Matrix\_Full2Diag is a subroutine that converts an unpacked MQC matrix to diagonal-packed.  
TODO: make tests for diagonal structure more efficient.

**Parameters**

<code>in, out</code>	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be diagonal packed.
----------------------	---------------	---

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.55 mqc\_matrix\_full2symm()**

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

**MQC\_Matrix\_Full2Symm** is a subroutine that converts an unpacked MQC matrix to symmetric-packed

**Purpose:**

MQC\_Matrix\_Full2Symm is a subroutine that converts an unpacked MQC matrix to symmetric-packed.  
 TODO: make tests for symmetry more efficient.

**Parameters**

<code>in, out</code>	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be symmetric packed.
----------------------	---------------	--

**Author**

L. M. Thompson

**Date**

2016

### 5.1.2.56 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 )
```

### 5.1.2.57 mqc\_matrix\_havecomplex()

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

**MQC\_Matrix\_HaveComplex** is a function used to indicate if an MQC matrix has an allocated complex matrix

#### Purpose:

MQC\_Matrix\_HaveComplex is a function that returns TRUE if an MQC matrix has an allocated complex matrix and FALSE if it does not.

#### Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

#### Author

L. M. Thompson

#### Date

2017

### 5.1.2.58 mqc\_matrix\_havediagonal()

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

**MQC\_Matrix\_HaveDiagonal** is a function used to indicate if an MQC matrix is stored diagonal-packed

#### Purpose:

MQC\_Matrix\_HaveDiagonal is a function that returns TRUE if an MQC matrix is stored diagonal-packed and FALSE if it is not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.59 mqc\_matrix\_havefull()**

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

**MQC\_Matrix\_HaveFull is a function used to indicate if an MQC matrix is stored unpacked**

**Purpose:**

MQC\_Matrix\_HaveFull is a function that returns TRUE if an MQC matrix is stored unpacked and FALSE if it is not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.60 mqc\_matrix\_haveinteger()**

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

**MQC\_Matrix\_HaveInteger** is a function used to indicate if an MQC matrix has an allocated integer matrix

**Purpose:**

MQC\_Matrix\_HaveInteger is a function that returns TRUE if an MQC matrix has an allocated integer matrix and FALSE if it does not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.61 mqc\_matrix\_havereal()**

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

**MQC\_Matrix\_HaveReal** is a function used to indicate if an MQC matrix has an allocated real matrix

**Purpose:**

MQC\_Matrix\_HaveReal is a function that returns TRUE if an MQC matrix has an allocated real matrix and FALSE if it does not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.62 mqc\_matrix\_havesymmetric()**

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

**MQC\_Matrix\_HaveSymmetric** is a function used to indicate if an MQC matrix is stored symmetric-packed

**Purpose:**

MQC\_Matrix\_HaveSymmetric is a function that returns TRUE if an MQC matrix is stored symmetric-packed and FALSE if it is not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.63 mqc\_matrix\_identity()**

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



**5.1.2.64 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 )
```

**5.1.2.65 mqc\_matrix\_inverse()**

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

**5.1.2.66 mqc\_matrix\_isallocated()**

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

**MQC\_Matrix\_isAllocate** is a function that returns the allocation status of a MQC\_Matrix variable

**Purpose:**

MQC\_Matrix\_isAllocate is a function that returns the allocation status of a MQC\_Matrix variable. The function returns TRUE if the matrix is allocated and FALSE if the matrix is not allocated.

**Parameters**

in, out	<i>Matrix</i>	Matrix is Class(MQC_Matrix) The MQC matrix to be tested.
---------	---------------	---

**Author**

L. M. Thompson

**Date**

2017

### 5.1.2.67 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

#### Purpose:

MQC\_Matrix\_Matrix\_At is a function that returns the matrix between rows (I,J) and columns (K,L) of an MQC\_Matrix Mat as an MQC\_Matrix Matrix. If I, J, K or L is negative, the (N-I+1)th index value is selected.

#### 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

### 5.1.2.68 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 )
```

## 5.1.2.69 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 )
```

**MQC\_Matrix\_Matrix\_Put** is a subroutine that writes a submatrix to the specified position of a MQC matrix

**Purpose:**

MQC\_Matrix\_Matrix\_Put is a subroutine that writes a submatrix to the specified position of a MQC matrix. The row and column specification are given as a vector where each vector must contain either zero or two non-zero integers to specify the range of elements that will be overwritten by the submatrix. If the value of an element specification is negative, it counts from the last element back. If the value of an element specification is zero, the whole row/column is specified.

**Parameters**

in, out	<i>Mat</i>	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The submatrix to overwrite at the specified elements of Mat.
in	<i>Rows</i>	Rows is Integer(kind=int64),Dimension(:) The specification of the rows to include in the subvector. If = [A,B]: output is subvector 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]: subvector of rows equivalent to [1,-1].
in	<i>Cols</i>	Cols is Integer(kind=int64),Dimension(:) The specification of the columns to include in the subvector. If = [A,B]: output is subvector 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]: subvector of columns equivalent to [1,-1].

**Author**

L. M. Thompson

Date

2017

### 5.1.2.70 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 )
```

### 5.1.2.71 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 )
```

### 5.1.2.72 mqc\_matrix\_rows()

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

**MQC\_Matrix\_Rows** is a function used to return the number of rows of an MQC matrix

**Purpose:**

MQC\_Matrix\_Rows is a function used to return the number of rows of an MQC matrix. If the matrix is NOT allocated, the number of rows is returned as 0.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

Date

2016

**5.1.2.73 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 )
```

**MQC\_Matrix\_Scalar\_At** is a function that returns the value of an element of a MQC matrix

Purpose:

MQC\_Matrix\_Scalar\_At is a function that returns the value of (I,J)th element of a MQC matrix as an MQC scalar. If I or J is negative, the (N-I+1)th index is selected.

Parameters

in	<i>Mat</i>	Mat is Class(MQC_Matrix) The MQC matrix to return the value of the (I,J)th element.
in	<i>I</i>	I is Integer(kind=int64) The row of the element in MQC matrix. If I>0 row count is from first index If I<0 row count is from last index.
in	<i>J</i>	J is Integer(kind=int64) The column of the element in MQC matrix. If J>0 row count is from first index If J<0 row count is from last index.

Author

X. Sheng

L. M. Thompson

Date

2017

#### 5.1.2.74 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 )
```

#### 5.1.2.75 mqc\_matrix\_set()

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

#### 5.1.2.76 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 )
```

#### 5.1.2.77 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 )
```

#### 5.1.2.78 mqc\_matrix\_symm2diag()

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

**MQC\_Matrix\_Symm2Diag** is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed

**Purpose:**

MQC\_Matrix\_Symm2Diag is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed.

TODO: make tests for diagonal structure more efficient.

## Parameters

in, out	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be diagonal packed.
---------	---------------	---

## Author

L. M. Thompson

## Date

2017

## 5.1.2.79 mqc\_matrix\_symm2full()

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

**MQC\_Matrix\_Symm2Full** is a subroutine that converts a symmetry-packed MQC matrix to unpacked

## Purpose:

MQC\_Matrix\_Symm2Full is a subroutine that converts a symmetry-packed MQC matrix to unpacked. The following options are available:

Option = 'symmetric' unpacks as if matrix is symmetric.  
 Option = 'antisymmetric' unpacks as if matrix is antisymmetric.  
 Option = 'hermitian' unpacks as if matrix is hermitian.  
 Option = 'antihermitian' unpacks as if matrix is antihermitian.

TODO: when different symm storage flags implemented, use these rather than an option

## Parameters

in, out	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to unpack.
in	<i>Option</i>	Option is Character(len=*),Optional = 'symmetric': Unpack as if matrix is symmetric = 'antisymmetric': Unpack as if matrix is antisymmetric = 'hermitian': Unpack as if matrix is hermitian = 'antihermitian': Unpack as if matrix is antihermitian

**Author**

L. M. Thompson

**Date**

2016, 2018

**5.1.2.80 mqc\_matrix\_symm2full\_func()**

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

**MQC\_Matrix\_Symm2Full\_Func** is a function that converts a symmetric- packed MQC matrix to unpacked

**Purpose:**

MQC\_Matrix\_Symm2Full\_Func is a function that converts a symmetric-packed MQC matrix to unpacked.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.
----	---------------	--

**Author**

X. Sheng

**Date**

2017

**5.1.2.81 mqc\_matrix\_symmetrize()**

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

**MQC\_Matrix\_Symmetrize** is a function that symmetrizes a MQC matrix

**Purpose:**

MQC\_Matrix\_Symmetrize is a function that symmetrizes a MQC matrix.  
TODO: options to antisymmetrize, hermitianize and antihermitianize.



## Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be symmetrized.
----	---------------	---

## Author

L. M. Thompson

## Date

2016

## 5.1.2.82 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Complex** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector

## Purpose:

MQC\_Matrix\_SymmMatrix\_Put\_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector.

## Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	<i>Symm↔ MatrixIn</i>	SymmMatrixIn is complex(kind=real64),dimension(:) Intrinsic complex vector to write as symmetric-packed matrix.

## Author

L. M. Thompson

Date

2017

### 5.1.2.83 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Integer** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector

**Purpose:**

MQC\_Matrix\_SymmMatrix\_Put\_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector.

**Parameters**

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	<i>SymmMatrixIn</i>	SymmMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as symmetric-packed matrix.

**Author**

H. P. Hratchian

Date

2017

### 5.1.2.84 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Real** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector

**Purpose:**

MQC\_Matrix\_SymmMatrix\_Put\_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector.

**Parameters**

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	<i>SymmMatrixIn</i>	SymmMatrixIn is real(kind=real64),dimension(:) Intrinsic real vector to write as symmetric-packed matrix.

**Author**

H. P. Hratchian

**Date**

2017

**5.1.2.85 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 )
```

**5.1.2.86 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 )
```

**5.1.2.87 mqc\_matrix\_test\_diagonal()**

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

**MQC\_Matrix\_Test\_Diagonal is a function that tests a MQC matrix to determine if it is diagonal**

**Purpose:**

MQC\_Matrix\_Test\_Diagonal is a function that tests a MQC matrix to determine if it is diagonal. The function returns TRUE if the matrix is diagonal and FALSE if the matrix is not diagonal.

**Parameters**

in	<i>Matrix</i>	Matrix is Class(mqc_matrix) The matrix to be tested.
----	---------------	---

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.88 mqc\_matrix\_test\_symmetric()**

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

**MQC\_Matrix\_Test\_Symmetric** is a function that tests a MQC matrix for symmetry

**Purpose:**

MQC\_Matrix\_Test\_Symmetric is a function that tests a MQC matrix for symmetry specified by optional argument, with the default test being for a symmetric matrix. Note that this function differs from haveSymmetric subroutine which tests how a matrix is packed. The following options are available:

1. Option = 'symmetric' tests for a symmetric matrix (default).
2. Option = 'antisymmetric' tests for an antisymmetric matrix.
3. Option = 'hermitian' tests for a hermitian matrix.
4. Option = 'antihermitian' tests for an antihermitian matrix.

**Parameters**

in	<i>Matrix</i>	Matrix is Class(mqc_matrix) The matrix to be tested for symmetry.
in	<i>Option</i>	Option is Character(len=*),Optional = 'symmetric': symmetric matrix test = 'antisymmetric': antisymmetric matrix test = 'hermitian': hermitian matrix test = 'antihermitian': antihermitian matrix test.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.89 mqc\_matrix\_trace()**

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

**5.1.2.90 mqc\_matrix\_transpose()**

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

**MQC\_Matrix\_Transpose is a function that returns the transpose of a MQC matrix**

**Purpose:**

MQC\_Matrix\_Transpose is a function that returns the transpose of a MQC matrix.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be transposed.
----	---------------	--

**Author**

L. M. Thompson

X. Sheng

**Date**

2016, 2017

### 5.1.2.91 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 )
```

**MQC\_Matrix\_Vector\_At** is a function that returns the subvector of an MQC matrix

#### Purpose:

MQC\_Matrix\_Vector\_At is a function that returns the subvector of an MQC matrix. The row and column specification are given as a vector where one vector must contain a single non-zero integer to specify the row or column of the subvector, and the other vector must be either zero or two non-zero integers to specify the range of elements that will form the subvector. If the value of an element specification is negative, it counts from the last element back. If the value of an element specification is zero, the whole row/column is specified.

#### Parameters

in	<i>Mat</i>	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.
in	<i>Rows</i>	Rows is Integer(kind=int64),Dimension(:) The specification of the rows to include in the subvector. If = [A,B]: output is subvector 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]: subvector of rows equivalent to [1,-1].
in	<i>Cols</i>	Cols is Integer(kind=int64),Dimension(:) The specification of the columns to include in the subvector. If = [A,B]: output is subvector 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]: subvector of columns equivalent to [1,-1].

#### Author

L. M. Thompson

#### Date

2017

## 5.1.2.92 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 )
```

**MQC\_Matrix\_Vector\_Put** is a subroutine that writes a subvector to the specified position of a MQC matrix

**Purpose:**

MQC\_Matrix\_Vector\_Put is a subroutine that writes a subvector to the specified position of a MQC matrix. The row and column specification are given as a vector where one vector must contain a single non-zero integer to specify the row or column where the subvector will be written, and the other vector must be either zero or two non-zero integers to specify the range of elements that will be overwritten by the subvector. If the value of an element specification is negative, it counts from the last element back. If the value of an element specification is zero, the whole row/column is specified.

**Parameters**

in, out	<i>Mat</i>	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The subvector to overwrite at the specified elements of Mat.
in	<i>Rows</i>	Rows is Integer(kind=int64),Dimension(:) The specification of the rows to include in the subvector. If = [A,B]: output is subvector 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]: subvector of rows equivalent to [1,-1].
in	<i>Cols</i>	Cols is Integer(kind=int64),Dimension(:) The specification of the columns to include in the subvector. If = [A,B]: output is subvector 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]: subvector of columns equivalent to [1,-1].

**Author**

L. M. Thompson

Date

2017

**5.1.2.93 mqc\_matrixmatrixdotproduct()**

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

**5.1.2.94 mqc\_matrixmatrixproduct()**

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

**MQC\_MatrixMatrixProduct** is a function that computes the element- wise product of two MQC matrices

Purpose:

MQC\_MatrixMatrixProduct is a function that computes the element-wise product of two MQC matrices.

Parameters

in	<i>MA</i>	MA is Type(MQC_Matrix) The first MQC matrix.
in	<i>MB</i>	MB is Type(MQC_Matrix) The second MQC matrix.

Author

H. P. Hratchian

Date

2017



**5.1.2.95 mqc\_matrixmatrixsubtract()**

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

**MQC\_MatrixMatrixSubtract** is a function that subtracts two MQC matrices

**Purpose:**

MQC\_MatrixMatrixSubtract is a function that subtracts two MQC matrices.

**Parameters**

in	<i>MA</i>	MA is Type(MQC_Matrix) The matrix that MB will be subtracted from.
in	<i>MB</i>	MB is Type(MQC_Matrix) The matrix that will be subtracted from MA.

**Author**

H. P. Hratchian

**Date**

2017

**5.1.2.96 mqc\_matrixmatrixsum()**

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

**MQC\_MatrixMatrixSum** is a function that sums two MQC matrices

**Purpose:**

MQC\_MatrixMatrixSum is a function that sums two MQC matrices.

**Parameters**

in	<i>MA</i>	MA is Type(MQC_Matrix) First MQC matrix to sum.
in	<i>MB</i>	MB is Type(MQC_Matrix) Second MQC matrix to sum.

**Author**

H. P. Hratchian

L. M. Thompson

**Date**

2017, 2018

**5.1.2.97 mqc\_matrixscalarproduct()**

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

**5.1.2.98 mqc\_matrixvectordotproduct()**

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

**5.1.2.99 mqc\_outer()**

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

**MQC\_Outer** is a function that returns the outer product of two MQC vectors

**Purpose:**

MQC\_Outer is a function that returns the outer product of two MQC vectors. The first vector should be a column vector, while the second vector should be a row vector.

## Parameters

in	<i>VA</i>	VA is Type(MQC_Vector) The MQC column vector.
in	<i>VB</i>	VB is Type(MQC_Vector) The MQC row vector.

## Author

X. Sheng

## Date

2017

## 5.1.2.100 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

#### 5.1.2.101 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

#### 5.1.2.102 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

**5.1.2.103 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

**5.1.2.104 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 )

```

**MQC\_Print\_Matrix\_Algebra1** is a subroutine used to print an MQC matrix

**Purpose:**

MQC\_Print\_Matrix\_Algebra1 is a subroutine used to print an MQC matrix.  
 Blank\_At\_Top and Blank\_At\_Bottom are optional logical arguments to print blank lines before or after output.

**Parameters**

in	<i>Matrix</i>	Matrix is Class(MQC_Matrix) 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 Matrix.
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

**5.1.2.105 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 )

```

**5.1.2.106 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.
Generated by Doxygen		
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

**5.1.2.107 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 )

```

**MQC\_Print\_Vector\_Algebra1** is a subroutine used to print an MQC vector

**Purpose:**

MQC\_Print\_Vector\_Algebra1 is a subroutine used to print an MQC vector.  
 Blank\_At\_Top and Blank\_At\_Bottom are optional logical arguments to print blank lines before or after output.

**Parameters**

in	<i>Vector</i>	Vector is Class(MQC_Vector) 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 Vector.
in	<i>Verbose</i>	Verbose is Logical,Optional Adds extra printing to output.
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

**5.1.2.108 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 )
```

**5.1.2.109 mqc\_r4tensor\_havecomplex()**

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

**5.1.2.110 mqc\_r4tensor\_haveinteger()**

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

**5.1.2.111 mqc\_r4tensor\_havereal()**

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

### 5.1.2.112 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 )
```

### 5.1.2.113 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 )
```

### 5.1.2.114 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

**5.1.2.115 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

### 5.1.2.116 `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

### 5.1.2.117 `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

## 5.1.2.118 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>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

#### 5.1.2.119 `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

#### 5.1.2.120 `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

## 5.1.2.121 mqc\_realvectorproduct()

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

**MQC\_RealVectorProduct** is a function that returns the product of an intrinsic real scalar and a MQC vector

## Purpose:

MQC\_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector.

## Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The real intrinsic to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.122 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

**5.1.2.123 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

**5.1.2.124 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

### 5.1.2.125 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	Scalar	Scalar is Type(MQC_Scalar) The argument of the function.
----	--------	---

#### Author

L. M. Thompson

#### Date

2019

### 5.1.2.126 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

## 5.1.2.127 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↔ In</i>	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.
----	-----------------------	--

## Author

L. M. Thompson

## Date

2018

### 5.1.2.128 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

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

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.130 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

**5.1.2.131 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

**5.1.2.132 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

**5.1.2.133 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

**5.1.2.134 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

**5.1.2.135 mqc\_scalar\_get\_random\_value()**

```

subroutine mqc_algebra::mqc_scalar_get_random_value (
    class(mqc_scalar) Scalar,
    integer, dimension(:), optional Seed,
    character(len=*), intent(in), optional Distribution )

```

**MQC\_Scalar\_Get\_Random\_Value** is a function that returns a random real value from a specified distribution

**Purpose:**

MQC\_Scalar\_Get\_Random\_Value is a function that returns a random real value from an optionally specified distribution. Note that the range of values varies by distribution. In addition, a seed can be specified for consistent greeneration of the same number. Default options are uniform distribution with random seed. The following options are available:

1. Distribution = 'uniform' uses a uniform distribution between 0 and 1.
2. Distribution = 'gaussian' uses a normal distribution with zero mean and unit variance obtained using the Box-Muller transformation
3. Distribution = 'exp' uses an exponential distribution (lambda=1.0) returning positive integers.
4. Distribution = 'exp01' uses an exponential distribution (lambda=8.0) that gives values in the range 0 and 1.

**Parameters**

in, out	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be filled.
in	<i>Seed</i>	Seed is integer,dimension(:),optional Integer array containing seed. Note in gfortran only the first two elements affect the value of the random number.
in	<i>Distribution</i>	Distribution is character(len=*),intent(in),optional Distribution of the function from which random number is selected. = 'uniform': uniform between 0 and 1 = 'gaussian': normal deviation with zero mean and unit variance = 'exp': exponential decay $y=\exp(-x)$ = 'exp01': exponential decay between 0 and 1 obtained using $\text{mod}(y,1.0)$ of $y=\exp(-8x)$ .



**Author**

X. Dong

L. M. Thompson

**Date**

2019

**5.1.2.136 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

**5.1.2.137 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( <code>MQC_Scalar</code> ) The <code>MQC_Scalar</code> to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.138 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( <code>MQC_Scalar</code> ) The <code>MQC_Scalar</code> to be tested.
----	---------------	--

**Author**

L. M. Thompson

Date

2017

### 5.1.2.139 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

### 5.1.2.140 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

**5.1.2.141 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

## 5.1.2.142 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

## 5.1.2.143 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.
Generated by Doxygen		

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.144 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>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

**5.1.2.145 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 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

**5.1.2.146 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>	Scalar1 is Type(MQC_Scalar) The base value.
in	<i>ComplexIn</i>	CompIn is Complex(kind=real64) The power value.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.147 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

**5.1.2.148 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

**5.1.2.149 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

**5.1.2.150 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

### 5.1.2.151 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(MQC_Scalar) The base value.
in	<i>Scalar2</i>	Scalar2 is Type(MQC_Scalar) The power value.

**Author**

L. M. Thompson

Date

2016

### 5.1.2.152 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 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

**5.1.2.153 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

## 5.1.2.154 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

**5.1.2.155 mqc\_scalargtreal()**

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

**MQC\_ScalarGTRReal is a function that returns TRUE if a MQC\_Scalar is greater than an intrinsic real**

**Purpose:**

MQC\_ScalarGTRReal 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>RealIn</i>	RealIn is Real(kind=int64) The intrinsic real that will be tested.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.156 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>IntegerIn</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

**5.1.2.157 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>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable denominator.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.158 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

**5.1.2.159 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>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

**5.1.2.160 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

**5.1.2.161 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

**5.1.2.162 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

### 5.1.2.163 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>RealIn</i>	RealIn is Real(kind=int64) The intrinsic real that will be tested.

#### Author

L. M. Thompson

#### Date

2019

### 5.1.2.164 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

## 5.1.2.165 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

**5.1.2.166 mqc\_scalarmatrixproduct()**

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

**5.1.2.167 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

## 5.1.2.168 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

## 5.1.2.169 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

**5.1.2.170 mqc\_scalarrealdive()**

```
type(mqc_scalar) function mqc_algebra::mqc_scalarrealdive (
    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 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

**5.1.2.171 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

**5.1.2.172 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

**5.1.2.173 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

#### 5.1.2.174 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

#### 5.1.2.175 mqc\_scalarvectordifference()

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

**MQC\_ScalarVectorDifference** is a function that subtracts an MQC scalar from all elements of an MQC vector

Purpose:

MQC\_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector.

**Parameters**

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC scalar to be subtracted from elements of the the MQC vector.
in	<i>Vector</i> ↔ <i>In</i>	VectorIn is Type(MQC_Vector) The MQC vector with elements from which ScalarIn will be subtracted.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.176 mqc\_scalarvectorproduct()**

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

**MQC\_ScalarVectorProduct** is a function that returns the product of a MQC scalar with a MQC vector

**Purpose:**

MQC\_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector.

**Parameters**

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

X. Sheng

A. D. Mahler

**Date**

2017, 2019

**5.1.2.177 mqc\_scalarvectorsum()**

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

**MQC\_ScalarVectorSum** is a function that adds an MQC scalar to all elements of an MQC vector

**Purpose:**

MQC\_VectorVectorSum is a function that adds an MQC scalar to all elements of an MQC vector.

**Parameters**

in	<i>Scalar↔ In</i>	ScalarIn is Type(MQC_Scalar) The MQC scalar to add to the MQC vector.
in	<i>Vector↔ In</i>	VectorIn is Type(MQC_Vector) The MQC vector with elements to sum with ScalarIn.

**Author**

L. M. Thompson

**Date**

2016

### 5.1.2.178 mqc\_set\_array2tensor()

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

### 5.1.2.179 mqc\_set\_array2vector\_complex()

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

**MQC\_Set\_Array2Vector\_Complex** is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

#### Purpose:

MQC\_Set\_Array2Vector\_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector.

#### Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Complex(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

#### Author

L. M. Thompson

#### Date

2017

**5.1.2.180 mqc\_set\_array2vector\_integer()**

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

**MQC\_Set\_Array2Vector\_Integer** is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

**Purpose:**

MQC\_Set\_Array2Vector\_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

**Parameters**

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Integer(kind=int64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

H. P. Hratchian

**Date**

2016

**5.1.2.181 mqc\_set\_array2vector\_real()**

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

**MQC\_Set\_Array2Vector\_Real** is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

**Purpose:**

MQC\_Set\_Array2Vector\_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector.

**Parameters**

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

H. P. Hratchian

**Date**

2016

**5.1.2.182 mqc\_set\_complexarray2matrix()**

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

**MQC\_Set\_ComplexArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array

**Purpose:**

MQC\_Set\_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the complex array.
in	<i>ArrayIn</i>	ArrayIn is Complex(kind=real64),Dimension(:,:) The complex array to be input into MatrixOut.



**Author**

L. M. Thompson

**Date**

2017

**5.1.2.183 mqc\_set\_integerarray2matrix()**

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

**MQC\_Set\_IntegerArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array

**Purpose:**

MQC\_Set\_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the integer array.
in	<i>ArrayIn</i>	ArrayIn is Integer(kind=int64),Dimension(:,:) The integer array to be input into MatrixOut.

**Author**

L. M. Thompson

**Date**

2016

### 5.1.2.184 mqc\_set\_matrix2complexarray()

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

**MQC\_Set\_Matrix2ComplexArray** is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix

#### Purpose:

MQC\_Set\_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix.

#### Parameters

in, out	<i>ArrayOut</i>	ArrayOut is Complex(kind=real64),Dimension(:,:),Allocatable The complex array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

#### Author

L. M. Thompson

#### Date

2017

### 5.1.2.185 mqc\_set\_matrix2integerarray()

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

**MQC\_Set\_Matrix2IntegerArray** is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix

#### Purpose:

MQC\_Set\_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix.

## Parameters

in, out	<i>ArrayOut</i>	ArrayOut is Integer(kind=int64),Dimension(:,:),Allocatable The integer array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

## Author

L. M. Thompson

## Date

2016

## 5.1.2.186 mqc\_set\_matrix2matrix()

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

**MQC\_Set\_Matrix2Matrix** is a subroutine that sets an MQC matrix equal to another MQC matrix

## Purpose:

MQC\_Set\_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix.

## Parameters

in, out	<i>MatrixOut</i>	MatrixOut is Class(MQC_Matrix) The MQC matrix to be set equal to the incoming MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Class(MQC_Matrix) The MQC matrix to be input into MatrixOut.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.187 mqc\_set\_matrix2realarray()**

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

**MQC\_Set\_Matrix2RealArray** is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix

**Purpose:**

MQC\_Set\_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Real(kind=real64),Dimension(:, :),Allocatable The real array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.188 mqc\_set\_realarray2matrix()**

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

**MQC\_Set\_RealArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array

**Purpose:**

MQC\_Set\_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the real array.
in	<i>ArrayIn</i>	ArrayIn is Real(kind=real64), Dimension(:, :) The real array to be input into MatrixOut.

**Author**

L. M. Thompson

**Date**

2016

**5.1.2.189 mqc\_set\_vector2complexarray()**

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

**MQC\_Set\_Vector2ComplexArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array

**Purpose:**

MQC\_Set\_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array.

**Parameters**

in, out	<i>ArrayOut</i>	<p>ArrayOut is Complex(kind=real64),Dimension(:)  The rank 1 intrinsic array which will receive the contents of MQC_Vector.</p>
in	<i>VectorIn</i>	<p>VectorIn is Type(MQC_Vector)  The MQC_Vector whose data will be output into the intrinsic array.</p>

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.190 mqc\_set\_vector2integerarray()**

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

**MQC\_Set\_Vector2IntegerArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array

**Purpose:**

MQC\_Set\_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array.

**Parameters**

in, out	<i>ArrayOut</i>	<p>ArrayOut is Integer(kind=int64),Dimension(:)  The rank 1 intrinsic array which will receive the contents of MQC_Vector.</p>
in	<i>VectorIn</i>	<p>VectorIn is Type(MQC_Vector)  The MQC_Vector whose data will be output into the intrinsic array.</p>

**Author**

H. P. Hratchian

**Date**

2016

**5.1.2.191 mqc\_set\_vector2realarray()**

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

**MQC\_Set\_Vector2RealArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array

**Purpose:**

MQC\_Set\_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array which will receive the contents of MQC_Vector.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC_Vector whose data will be output into the intrinsic array.

**Author**

H. P. Hratchian

**Date**

2016

### 5.1.2.192 mqc\_set\_vector2vector()

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

**MQC\_Set\_Vector2Vector** is a subroutine that sets a MQC vector equal to another MQC vector

#### Purpose:

MQC\_Set\_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector.

#### Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to VectorIn.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC vector whose contents will be copied to VectorOut.

#### Author

H. P. Hratchian

L. M. Thompson

#### Date

2016

### 5.1.2.193 mqc\_vector2diagmatrix()

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

**MQC\_Vector2DiagMatrix** is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector

#### Purpose:

MQC\_Vector2DiagMatrix is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector.



## Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) MQC vector defining diagonal elements of output matrix.
----	---------------	--

## Author

X. Sheng

## Date

2017

## 5.1.2.194 mqc\_vector\_abs()

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

**MQC\_Vector\_Abs** is a function that returns the absolute value of all elements of an MQC vector

## Purpose:

MQC\_Vector\_Sqrt is a function that returns the absolute value of all elements of an MQC vector.

## Parameters

in	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
----	----------	--

## Author

L. M. Thompson

## Date

2019

### 5.1.2.195 mqc\_vector\_argsort()

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

**MQC\_Vector\_Argsort** is a function that returns the indices of an an MQC vector sorted from low to high

#### Purpose:

MQC\_Vector\_Argsort is a function that returns the indices of an MQC vector sorted from low to high.

#### Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
----	---------------	---

#### Author

L. M. Thompson

#### Date

2019

### 5.1.2.196 mqc\_vector\_cast\_complex()

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

**MQC\_vector\_cast\_complex** is a function that converts an MQC vector to its complex space

#### Purpose:

MQC\_vector\_cast\_complex is a function that converts an MQC vector to its complex space.

#### Parameters

in	<i>VA</i>	VA is Class(MQC_Vector) The MQC vector to convert.
----	-----------	---

## Author

L. M. Thompson

## Date

2017

**5.1.2.197 mqc\_vector\_cast\_integer()**

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

**MQC\_vector\_cast\_integer** is a function that converts an MQC vector to its integer space

**Purpose:**

MQC\_vector\_cast\_integer is a function that converts an MQC vector to its integer space.

**Parameters**

in	VA	VA is Type(MQC_Vector) The MQC vector to convert.
----	----	--

## Author

L. M. Thompson

## Date

2019

**5.1.2.198 mqc\_vector\_cast\_real()**

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

**MQC\_vector\_cast\_real** is a function that converts an MQC vector to its real space

**Purpose:**

MQC\_vector\_cast\_real is a function that converts an MQC vector to its real space.

**Parameters**

in	<i>VA</i>	VA is Class(MQC_Vector) The MQC vector to convert.
----	-----------	---

**Author**

X. Sheng

**Date**

2017

**5.1.2.199 mqc\_vector\_cmplx()**

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

**MQC\_Vector\_Cmplx** is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

**Purpose:**

MQC\_Vector\_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector.

**Parameters**

in	<i>Vector1</i>	Vector1 is Type(MQC_Vector) The MQC vector containing the real part.
in	<i>Vector2</i>	Vector2 is Type(MQC_Vector) The MQC vector containing the imaginary part.

**Author**

L. M. Thompson

Date

2019

### 5.1.2.200 mqc\_vector\_complex\_imagpart()

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

**MQC\_Vector\_Complex\_ImagPart** is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

Purpose:

MQC\_Vector\_Complex\_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector.

Parameters

in	A	
		A is Class(MQC_Vector) The name of the MQC_Vector variable.

Author

L. M. Thompson

Date

2019

### 5.1.2.201 mqc\_vector\_complex\_realpart()

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

**MQC\_Vector\_Complex\_RealPart** is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

Purpose:

MQC\_Vector\_Complex\_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector.

**Parameters**

in	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
----	----------	--

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.202 mqc\_vector\_conjugate\_transpose()**

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

**MQC\_Vector\_Conjugate\_Transpose** is a function that returns the conjugate transpose of an MQC vector

**Purpose:**

MQC\_Vector\_Conjugate\_Transpose is a function that returns the conjugate transpose of an MQC vector.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC vector to conjugate transpose.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2017

### 5.1.2.203 mqc\_vector\_copy\_complex2int()

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

**MQC\_Vector\_Copy\_Complex2Int** is a subroutine that copies a complex MQC\_Vector into its integer vector

**Purpose:**

MQC\_Vector\_Copy\_Complex2Int is a subroutine that copies a complex MQC\_Vector into its integer vector.

**Parameters**

in, out	Vector	
		Vector is Type(MQC_Vector) The MQC_Vector to be tested.

**Author**

L. M. Thompson

**Date**

2017

### 5.1.2.204 mqc\_vector\_copy\_complex2real()

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

**MQC\_Vector\_Copy\_Complex2Real** is a subroutine that copies a complex MQC\_Vector into its real vector

**Purpose:**

MQC\_Vector\_Copy\_Complex2Real is a subroutine that copies a complex MQC\_Vector into its real vector.

**Parameters**

in, out	Vector	
		Vector is Type(MQC_Vector) The MQC_Vector to be tested.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.205 mqc\_vector\_copy\_int2complex()**

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

**MQC\_Vector\_Copy\_Int2Complex** is a subroutine that copies an integer MQC\_Vector into its complex vector

**Purpose:**

MQC\_Vector\_Copy\_Int2Complex is a subroutine that copies an integer MQC\_Vector into its complex vector.

**Parameters**

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.206 mqc\_vector\_copy\_int2real()**

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

**MQC\_Vector\_Copy\_Int2Real** is a subroutine that copies an integer MQC\_Vector into its real vector

**Purpose:**

MQC\_Vector\_Copy\_Int2Real is a subroutine that copies an integer MQC\_Vector into its real vector.



## Parameters

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

## Author

H. P. Hratchian

## Date

2016

**5.1.2.207 mqc\_vector\_copy\_real2complex()**

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

**MQC\_Vector\_Copy\_Real2Complex** is a subroutine that copies a real MQC\_Vector into its complex vector

## Purpose:

MQC\_Vector\_Copy\_Real2Complex is a subroutine that copies a real MQC\_Vector into its complex vector.

## Parameters

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

## Author

L. M. Thompson

## Date

2017

### 5.1.2.208 mqc\_vector\_copy\_real2int()

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

**MQC\_Vector\_Copy\_Real2Int** is a subroutine that copies a real MQC\_Vector into its integer vector

#### Purpose:

MQC\_Vector\_Copy\_Real2Int is a subroutine that copies a real MQC\_Vector into its integer vector.

#### Parameters

in, out	Vector	
		Vector is Type(MQC_Vector) The MQC_Vector to be tested.

#### Author

L. M. Thompson

#### Date

2016

### 5.1.2.209 mqc\_vector\_havecomplex()

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

**MQC\_Vector\_HaveComplex** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector

#### Purpose:

MQC\_Vector\_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector.

#### Parameters

in, out	Vector	
		Vector is Type(MQC_Vector) The MQC_Vector to be tested.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.210 mqc\_vector\_haveinteger()**

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

**MQC\_Vector\_HaveInteger** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector

**Purpose:**

MQC\_Vector\_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector.

**Parameters**

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

**Author**

H. P. Hratchian

**Date**

2016

**5.1.2.211 mqc\_vector\_havereal()**

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

**MQC\_Vector\_HaveReal** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector

**Purpose:**

`MQC_Vector_HaveReal` is a function that returns `TRUE` or `FALSE` indicating whether the MQC vector has an allocated real vector.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
----	---------------	--

## Author

H. P. Hratchian

## Date

2016

## 5.1.2.212 mqc\_vector\_initialize()

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

**MQC\_Vector\_Initialize** is a subroutine that initializes a MQC vector

## Purpose:

MQC\_Vector\_Initialize is a subroutine that initializes a MQC vector. Default element values are 0.0, or otherwise vector can be initialized with optional argument.

## Parameters

in, out	<i>Vector</i>	Vector is Class(MQC_Vector) The MQC_Vector to initialize.
in	<i>Length</i>	Length is Integer(kind=int64) The length to initialize vector
in	<i>Scalar</i>	Scalar is Class(*),Optional Value to set each element of Vector. If not present, the value is set to 0.0. Can be of type integer, real, complex or MQC_Scalar.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.213 mqc\_vector\_isallocated()**

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

**MQC\_Vector\_isAllocated** is a function that returns TRUE is an MQC vector is allocated and FALSE if it is not

**Purpose:**

MQC\_Vector\_isAllocated is a function that returns TRUE is an MQC vector is allocated and FALSE if it is not.

**Parameters**

in, out	<i>Vector</i>	<p>Vector is Class(MQC_Vector) The name of the MQC_Vector variable.</p>
---------	---------------	---

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.214 mqc\_vector\_iscolumn()**

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

**MQC\_Vector\_IsColumn** is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector

**Purpose:**

MQC\_Vector\_IsColumn is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector.

## Parameters

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

## Author

L. M. Thompson

## Date

2016

## 5.1.2.215 mqc\_vector\_maxloc()

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

**MQC\_Vector\_MaxLoc** is a function that returns the index of the largest value in an MQC vector

## Purpose:

MQC\_Vector\_MaxLoc is a function that returns the index of the largest value in an MQC vector.

## Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
----	---------------	---

## Author

L. M. Thompson

## Date

2019

### 5.1.2.216 mqc\_vector\_maxval()

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

**MQC\_Vector\_MaxVal** is a function that returns the largest value in an MQC vector

#### Purpose:

MQC\_Vector\_MaxVal is a function that returns the largest value in an MQC vector.

#### Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
----	---------------	---

#### Author

L. M. Thompson

#### Date

2017

### 5.1.2.217 mqc\_vector\_minloc()

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

**MQC\_Vector\_MinLoc** is a function that returns the index of the smallest value in an MQC vector

#### Purpose:

MQC\_Vector\_MinLoc is a function that returns the index of the smallest value in an MQC vector.

#### Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
----	---------------	---



**Author**

L. M. Thompson

**Date**

2019

**5.1.2.218 mqc\_vector\_minval()**

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

**MQC\_Vector\_MinVal** is a function that returns the smallest value in an MQC vector

**Purpose:**

MQC\_Vector\_MinVal is a function that returns the smallest value in an MQC vector.

**Parameters**

in	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
----	---------------	---

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.219 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 )
```

**MQC\_Vector\_Norm** is a function that returns the norm of an MQC vector

**Purpose:**

MQC\_Vector\_Norm is a function that returns the norm of an MQC vector. The following options are available:

1. methodIn = 'M' uses the maximum absolute value `max(abs(A(i)))`.
2. methodIn = '1' uses the one norm.
3. methodIn = 'I' uses the infinity norm.
4. methodIn = 'F' uses the Frobenius norm (default).

**Parameters**

in, out	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
in	<i>MethodIn</i>	MethodIn is Character(len=1) = 'M': <code>max(abs(A(i)))</code> = '1': one norm = 'I': infinity norm = 'F': Frobenius norm.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.220 mqc\_vector\_pop()**

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

**MQC\_Vector\_Pop** is a function that removes a value from the end of a MQC vector and returns it

**Purpose:**

MQC\_Vector\_Pop is a function that removes a value from the end of a MQC vector and returns it.

## Parameters

in, out	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
---------	---------------	---

## Author

L. M. Thompson

## Date

2017

## 5.1.2.221 mqc\_vector\_power()

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

**MQC\_Vector\_Power** is a function that returns the value of all elements of an MQC vector raised to a power

## Purpose:

MQC\_Vector\_Power is a function that returns the value of all elements of an MQC vector raised to a power. The power can be integer, real, complex or an MQC scalar.

## Parameters

in	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
in	<i>P</i>	P is Class(*) The power to raise elements of the MQC vector.

## Author

L. M. Thompson

Date

2019

### 5.1.2.222 mqc\_vector\_push()

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

**MQC\_Vector\_Push** is a function that adds a value to the end of a MQC vector

**Purpose:**

MQC\_Vector\_Push is a function that adds a value to the end of a MQC vector.

**Parameters**

in, out	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The value that will be added to the end of Vector.

**Author**

L. M. Thompson

Date

2017

### 5.1.2.223 mqc\_vector\_scalar\_at()

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

**MQC\_Vector\_Scalar\_At** is a function that returns the ith element of a MQC vector as an MQC scalar

**Purpose:**

MQC\_Vector\_Scalar\_At is a function that returns the ith element of a MQC vector as an MQC scalar. If the location of the element is negative the it is counted from the end of MQC vector.

## Parameters

in	<i>Vec</i>	Vec is Class(MQC_Vector) The MQC_Vector to extract the Ith element.
in	<i>I</i>	I is Integer(kind=int64) The location of the element in Vec to return. If I is negative it is counted from the last element of Vec.

## Author

X. Sheng

## Date

2017

## 5.1.2.224 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 )
```

**MQC\_Vector\_Scalar\_Increment** is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar

## Purpose:

MQC\_Vector\_Scalar\_Increment is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar. If the location of the element is negative then it is counted from the end of MQC vector.

## Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) The MQC_Vector to update at the Ith element.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to update the Ith element.
in	<i>I</i>	I is Integer(kind=int64) The location of the element in Vector to update. If I is negative it is counted from the last element of Vector.
Generated by Doxygen		

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.225 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 )
```

**MQC\_Vector\_Scalar\_Put** is a subroutine that updates the value of the *ith* element of a MQC vector with the value of a MQC scalar

**Purpose:**

MQC\_Vector\_Scalar\_Put is a subroutine that updates the value of the *ith* element of a MQC vector with the value of a MQC scalar. If the location of the element is negative then it is counted from the end of MQC vector.

**Parameters**

in	<i>Vector</i>	Vector is Class(MQC_Vector) The MQC_Vector to update at the <i>Ith</i> element.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to update Vector at element <i>I</i> .
in	<i>I</i>	<i>I</i> is Integer(kind=int64) The location of the element in Vector to update. If <i>I</i> is negative it is counted from the last element of Vector.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.226 mqc\_vector\_shift()**

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

**MQC\_Vector\_Shift** is a function that removes a value from the beginning of a MQC vector and returns it

**Purpose:**

MQC\_Vector\_Pop is a function that removes a value from the beginning of a MQC vector and returns it.

**Parameters**

in,out	Vector	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
--------	--------	---

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.227 mqc\_vector\_sort()**

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

**MQC\_Vector\_Sort** is a function that returns an MQC vector sorted from low to high unless optional index order is present

**Purpose:**

MQC\_Vector\_Sort is a function that returns an MQC vector sorted from low to high unless optional argument idx is present which gives the new order of the vector.

**Parameters**

<code>in, out</code>	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
<code>in</code>	<i>idx</i>	idx is Type(MQC_Vector), Optional The new order of indices after sort.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.228 mqc\_vector\_sqrt()**

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

**MQC\_Vector\_Sqrt** is a function that returns the square root of all elements of an MQC vector

**Purpose:**

MQC\_Vector\_Sqrt is a function that returns the square root of all elements of an MQC vector.

**Parameters**

<code>in</code>	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
-----------------	----------	--

**Author**

L. M. Thompson

**Date**

2019



## 5.1.2.229 mqc\_vector\_transpose()

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

**MQC\_Vector\_Transpose** is a function that returns the transpose of an MQC vector

**Purpose:**

MQC\_Vector\_Transpose is a function that returns the transpose of an MQC vector.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC vector to transpose.
----	---------------	--

**Author**

H. P. Hratchian

**Date**

2016

## 5.1.2.230 mqc\_vector\_unshift()

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

**MQC\_Vector\_Unshift** is a function that adds a value to the beginning of a MQC vector

**Purpose:**

MQC\_Vector\_Unshift is a function that adds a value to the beginning of a MQC vector.

**Parameters**

in, out	<i>Vector</i>	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
in	<i>Scalar</i>	
Generated by Doxygen		Scalar is Type(MQC_Scalar) The value that will be added to the beginning of Vector.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.231 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 )

```

**MQC\_Vector\_Vector\_At** is a function that returns the vector at the specified subvector of MQC\_Vector

**Purpose:**

MQC\_Vector\_Vector\_At is a function that returns the vector at the specified subvector of MQC\_Vector. Negative values of I or J indicate that counting proceeds from the last element rather than the first as with positive numbers.

**Parameters**

in	<i>Vec</i>	Vec is Class(MQC_Vector) The MQC_Vector from which the subvector will be extracted.
in	<i>I</i>	I is Integer(kind=int64) The location of the first subvector element in Vec. If I is negative it is counted from the last element of Vec.
in	<i>J</i>	J is Integer(kind=int64) The location of the last subvector element in Vec. If J is negative it is counted from the last element of Vec.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.232 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 )
```

**MQC\_Vector\_Vector\_Put** is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector

**Purpose:**

MQC\_Vector\_Vector\_Put is a subroutine that updates the values of a subvector of a MQC vector starting at element *I* with the values of a MQC vector. Negative *I* counts from the end of vector.

**Parameters**

in	<i>Vector</i>	Vector is Class(MQC_Vector) The MQC_Vector to update starting at the <i>I</i> th element.
in	<i>Vector</i> ↔ <i>In</i>	VectorIn is Class(MQC_Vector) The subvector with values to insert into Vector.
in	<i>I</i>	<i>I</i> is Integer(kind=int64) The location of the first subvector element in Vector to update. If <i>I</i> is negative it is counted from the last element of Vector.

**Author**

L. M. Thompson

**Date**

2017

**5.1.2.233 mqc\_vectorcomplexdivide()**

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

**MQC\_VectorComplexDivide** is a function that returns a MQC vector divided by an intrinsic complex scalar

**Purpose:**

MQC\_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>Comp↔ In</i>	CompIn is Complex(kind=comp64) The intrinsic complex scalar to divide by.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.234 mqc\_vectorcomplexproduct()**

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

**MQC\_VectorComplexProduct** is a function that returns the product of a MQC vector and an intrinsic complex scalar

**Purpose:**

MQC\_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>Comp↔ In</i>	CompIn is Complex(kind=real64) The integer complex to multiply.

## Author

L. M. Thompson

## Date

2019

## 5.1.2.235 mqc\_vectorintegerdivide()

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

**MQC\_VectorIntegerDivide** is a function that returns a MQC vector divided by an intrinsic integer scalar

## Purpose:

MQC\_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer scalar to divide by.

## Author

L. M. Thompson

## Date

2019

### 5.1.2.236 mqc\_vectorintegerproduct()

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

**MQC\_VectorIntegerProduct** is a function that returns the product of a MQC vector and an intrinsic integer scalar

#### Purpose:

MQC\_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar.

#### Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The integer intrinsic to multiply.

#### Author

L. M. Thompson

#### Date

2019

### 5.1.2.237 mqc\_vectormatrixdotproduct()

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

### 5.1.2.238 mqc\_vectorrealdive()

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

**MQC\_VectorRealDivide** is a function that returns a MQC vector divided by an intrinsic real integer

**Purpose:**

MQC\_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>Real↵ In</i>	RealIn is Real(kind=real64) The intrinsic real scalar to divide by.

**Author**

L. M. Thompson

**Date**

2019

**5.1.2.239 mqc\_vectorrealproduct()**

```

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

```

**MQC\_VectorRealProduct** is a function that returns the product of a MQC vector and an intrinsic real scalar

**Purpose:**

MQC\_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>Real↵ In</i>	RealIn is Real(kind=real64) The real intrinsic to multiply.



**Author**

L. M. Thompson

**Date**

2019

**5.1.2.240 mqc\_vectorscalardivide()**

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

**MQC\_VectorScalarDivide** is a function that returns a MQC vector divided by a MQC scalar

**Purpose:**

MQC\_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to divide by.

**Author**

A. D. Mahler

**Date**

2019

### 5.1.2.241 mqc\_vectorscalarproduct()

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

**MQC\_VectorScalarProduct** is a function that returns the product of a MQC vector with a MQC scalar

#### Purpose:

MQC\_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar.

#### Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.

#### Author

X. Sheng

#### Date

2017

### 5.1.2.242 mqc\_vectorvectordifference()

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

**MQC\_VectorVectorDifference** is a function that subtracts two MQC vectors and stores them in another MQC vector

#### Purpose:

MQC\_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector.

## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector from which the second will be subtracted.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector that will be subtracted from the first.

## Author

L. M. Thompson

## Date

2016

## 5.1.2.243 mqc\_vectorvectordotproduct()

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

**MQC\_VectorVectorDotProduct** is a function that returns the dot product of two MQC vectors

## Purpose:

MQC\_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors. The first vector should be a row vector, while the second vector should be a column vector. The vectors should be of the same length.

## Parameters

in	<i>Vector1</i>	Vector1 is Type(MQC_Vector) The MQC row vector.
in	<i>Vector2</i>	Vector2 is Type(MQC_Vector) The MQC column vector.

**Author**

H. P. Hratchian  
L. M. Thompson

**Date**

2016

**5.1.2.244 mqc\_vectorvectorsum()**

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

**MQC\_VectorVectorSum** is a function that adds two MQC vectors and stores them in another MQC vector

**Purpose:**

MQC\_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector.

**Parameters**

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector that will be summed.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector that will be summed.

**Author**

L. M. Thompson

**Date**

2016

## 5.1.2.245 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 )
```

**SymIndexHash** is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor

**Purpose:**

SymIndexHash is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor. If a matrix is tested, it is assumed lower-triangular row-wise stored.

**Parameters**

in	<i>I</i>	I is integer(kind=int64) The first index of the matrix/rank-4 tensor.
in	<i>J</i>	I is integer(kind=int64) The second index of the matrix/rank-4 tensor.
in	<i>K</i>	K is integer(kind=int64),Optional The third index of the rank-4 tensor. This argument is only required when taking hash of a rank-4 tensor.
in	<i>L</i>	L is integer(kind=int64),Optional The fourth index of the rank-4 tensor. This argument is only required when taking hash of a rank-4 tensor.

**Author**

X. Sheng  
L. M. Thompson

**Date**

2017

## 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 nelec,
    integer(kind=int64), optional multi,
    integer(kind=int64), dimension(:), optional, allocatable elist )
```

#### 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

Takes the absolute value

#### 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)  
*MQC\_Vector\_Abs is a function that returns the absolute value of all elements of an MQC vector*

#### 6.1.1 Detailed Description

Takes the absolute value

#### 6.1.2 Member Function/Subroutine Documentation

##### 6.1.2.1 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	<i>Scalar</i>	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested.
----	---------------	---

**Author**

A. Mahler

**Date**

2018

**6.1.2.2 mqc\_vector\_abs()**

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

**MQC\_Vector\_Abs** is a function that returns the absolute value of all elements of an MQC vector

**Purpose:**

MQC\_Vector\_Sqrt is a function that returns the absolute value of all elements of an MQC vector.

**Parameters**

in	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
----	----------	--

**Author**

L. M. Thompson

**Date**

2019

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

- src/[mqc\\_algebra.F03](#)



## 6.2 mqc\_algebra::acos Interface Reference

Returns the arccosine

### 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 Detailed Description

Returns the arccosine

### 6.2.2 Member Function/Subroutine Documentation

#### 6.2.2.1 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	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](#)

## 6.3 mqc\_algebra::aimag Interface Reference

Returns the imaginary part

### 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)  
*MQC\_Vector\_Complex\_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector*

### 6.3.1 Detailed Description

Returns the imaginary part

### 6.3.2 Member Function/Subroutine Documentation

#### 6.3.2.1 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

## 6.3.2.2 mqc\_vector\_complex\_imagpart()

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

**MQC\_Vector\_Complex\_ImagPart** is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

## Purpose:

MQC\_Vector\_Complex\_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector.

## Parameters

in	<i>A</i>	A is Class(MQC_Vector) The name of the MQC_Vector variable.
----	----------	--

## Author

L. M. Thompson

## Date

2019

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

- src/[mqc\\_algebra.F03](#)

## 6.4 mqc\_algebra::asin Interface Reference

Returns the arcsine

### 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 Detailed Description

Returns the arcsine

### 6.4.2 Member Function/Subroutine Documentation

#### 6.4.2.1 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

Assigns a variable to the value of another

### 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)  
*MQC\_Set\_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector*
- subroutine [mqc\\_set\\_vector2integerarray](#) (ArrayOut, VectorIn)  
*MQC\_Set\_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array*
- subroutine [mqc\\_set\\_vector2realarray](#) (ArrayOut, VectorIn)  
*MQC\_Set\_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array*
- subroutine [mqc\\_set\\_vector2complexarray](#) (ArrayOut, VectorIn)  
*MQC\_Set\_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array*
- subroutine [mqc\\_set\\_array2vector\\_integer](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector*
- subroutine [mqc\\_set\\_array2vector\\_real](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector*
- subroutine [mqc\\_set\\_array2vector\\_complex](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector*
- subroutine [mqc\\_set\\_matrix2matrix](#) (MatrixOut, MatrixIn)  
*MQC\_Set\_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix*

- subroutine [mqc\\_set\\_matrix2integerarray](#) (ArrayOut, MatrixIn)  
***MQC\_Set\_Matrix2IntegerArray** is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix*
- subroutine [mqc\\_set\\_matrix2realarray](#) (ArrayOut, MatrixIn)  
***MQC\_Set\_Matrix2RealArray** is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix*
- subroutine [mqc\\_set\\_matrix2complexarray](#) (ArrayOut, MatrixIn)  
***MQC\_Set\_Matrix2ComplexArray** is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix*
- subroutine [mqc\\_set\\_integerarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_IntegerArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array*
- subroutine [mqc\\_set\\_realarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_RealArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array*
- subroutine [mqc\\_set\\_complexarray2matrix](#) (MatrixOut, ArrayIn)  
***MQC\_Set\_ComplexArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array*
- subroutine [mqc\\_set\\_array2tensor](#) (TensorOut, ArrayIn)

### 6.5.1 Detailed Description

Assigns a variable to the value of another

### 6.5.2 Member Function/Subroutine Documentation

#### 6.5.2.1 [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**

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

**Author**

L. M. Thompson

**Date**

2017

**6.5.2.2 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

**6.5.2.3 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**

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

**Author**

L. M. Thompson

**Date**

2016

**6.5.2.4 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**

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



**Author**

L. M. Thompson

**Date**

2017

**6.5.2.5 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

**6.5.2.6 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

**6.5.2.7 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

**6.5.2.8 mqc\_set\_array2tensor()**

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

**6.5.2.9 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 )
```

**MQC\_Set\_Array2Vector\_Complex** is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

**Purpose:**

MQC\_Set\_Array2Vector\_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector.

**Parameters**

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Complex(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

L. M. Thompson

Date

2017

**6.5.2.10 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 )
```

**MQC\_Set\_Array2Vector\_Integer** is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

Purpose:

MQC\_Set\_Array2Vector\_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Integer(kind=int64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

Author

H. P. Hratchian

Date

2016

**6.5.2.11 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 )
```

**MQC\_Set\_Array2Vector\_Real** is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

**Purpose:**

MQC\_Set\_Array2Vector\_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector.

**Parameters**

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

H. P. Hratchian

**Date**

2016

**6.5.2.12 mqc\_set\_complexarray2matrix()**

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

**MQC\_Set\_ComplexArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array

**Purpose:**

MQC\_Set\_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the complex array.
in	<i>ArrayIn</i>	ArrayIn is Complex(kind=real64),Dimension(:,:) The complex array to be input into MatrixOut.

**Author**

L. M. Thompson

**Date**

2017

**6.5.2.13 mqc\_set\_integerarray2matrix()**

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

**MQC\_Set\_IntegerArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array

**Purpose:**

MQC\_Set\_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the integer array.
in	<i>ArrayIn</i>	ArrayIn is Integer(kind=int64),Dimension(:,:) The integer array to be input into MatrixOut.

**Author**

L. M. Thompson

**Date**

2016

**6.5.2.14 mqc\_set\_matrix2complexarray()**

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

**MQC\_Set\_Matrix2ComplexArray** is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix

**Purpose:**

MQC\_Set\_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Complex(kind=real64),Dimension(:,:),Allocatable The complex array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

**Author**

L. M. Thompson

**Date**

2017

**6.5.2.15 mqc\_set\_matrix2integerarray()**

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

**MQC\_Set\_Matrix2IntegerArray** is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix

**Purpose:**

MQC\_Set\_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Integer(kind=int64),Dimension(:,:),Allocatable The integer array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

**Author**

L. M. Thompson

**Date**

2016

**6.5.2.16 mqc\_set\_matrix2matrix()**

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

**MQC\_Set\_Matrix2Matrix** is a subroutine that sets an MQC matrix equal to another MQC matrix

**Purpose:**

MQC\_Set\_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix.

**Parameters**

in, out	<i>MatrixOut</i>	MatrixOut is Class(MQC_Matrix) The MQC matrix to be set equal to the incoming MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Class(MQC_Matrix) The MQC matrix to be input into MatrixOut.



**Author**

L. M. Thompson

**Date**

2016

**6.5.2.17 mqc\_set\_matrix2realarray()**

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

**MQC\_Set\_Matrix2RealArray** is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix

**Purpose:**

MQC\_Set\_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Real(kind=real64),Dimension(:, :),Allocatable The real array to be set equal to the MQC matrix.
in	<i>MatrixIn</i>	MatrixIn is Type(MQC_Matrix) The MQC matrix to be input into ArrayOut.

**Author**

L. M. Thompson

**Date**

2016

### 6.5.2.18 mqc\_set\_realarray2matrix()

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

**MQC\_Set\_RealArray2Matrix** is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array

#### Purpose:

MQC\_Set\_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array.

#### Parameters

in, out	<i>MatrixOut</i>	MatrixOut is Type(MQC_Matrix) The MQC matrix to be set equal to the real array.
in	<i>ArrayIn</i>	ArrayIn is Real(kind=real64),Dimension(:, :) The real array to be input into MatrixOut.

#### Author

L. M. Thompson

#### Date

2016

### 6.5.2.19 mqc\_set\_vector2complexarray()

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

**MQC\_Set\_Vector2ComplexArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array

#### Purpose:

MQC\_Set\_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array.

## Parameters

in, out	<i>ArrayOut</i>	ArrayOut is Complex(kind=real64),Dimension(:) The rank 1 intrinsic array which will receive the contents of MQC_Vector.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC_Vector whose data will be output into the intrinsic array.

## Author

L. M. Thompson

## Date

2017

## 6.5.2.20 mqc\_set\_vector2integerarray()

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

**MQC\_Set\_Vector2IntegerArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array

## Purpose:

MQC\_Set\_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array.

## Parameters

in, out	<i>ArrayOut</i>	ArrayOut is Integer(kind=int64),Dimension(:) The rank 1 intrinsic array which will receive the contents of MQC_Vector.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC_Vector whose data will be output into the intrinsic array.

**Author**

H. P. Hratchian

**Date**

2016

**6.5.2.21 mqc\_set\_vector2realarray()**

```

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

```

**MQC\_Set\_Vector2RealArray** is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array

**Purpose:**

MQC\_Set\_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array.

**Parameters**

in, out	<i>ArrayOut</i>	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array which will receive the contents of MQC_Vector.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC_Vector whose data will be output into the intrinsic array.

**Author**

H. P. Hratchian

**Date**

2016

### 6.5.2.22 mqc\_set\_vector2vector()

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

**MQC\_Set\_Vector2Vector** is a subroutine that sets a MQC vector equal to another MQC vector

#### Purpose:

MQC\_Set\_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector.

#### Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to VectorIn.
in	<i>VectorIn</i>	VectorIn is Type(MQC_Vector) The MQC vector whose contents will be copied to VectorOut.

#### Author

H. P. Hratchian  
L. M. Thompson

#### Date

2016

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, integralln)
- 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

Returns the arctangent

### 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 Detailed Description

Returns the arctangent

### 6.7.2 Member Function/Subroutine Documentation

#### 6.7.2.1 `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

Returns the arctangent accounting for circle quadrant

### 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 Detailed Description

Returns the arctangent accounting for circle quadrant

### 6.8.2 Member Function/Subroutine Documentation

#### 6.8.2.1 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

Defines a complex number

### 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)  
***MQC\_Vector\_Cmplx** is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector*

### 6.9.1 Detailed Description

Defines a complex number

### 6.9.2 Member Function/Subroutine Documentation

#### 6.9.2.1 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

## 6.9.2.2 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 )
```

**MQC\_Vector\_Cmplx** is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

## Purpose:

MQC\_Vector\_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector.

## Parameters

in	<i>Vector1</i>	Vector1 is Type(MQC_Vector) The MQC vector containing the real part.
in	<i>Vector2</i>	Vector2 is Type(MQC_Vector) The MQC vector containing the imaginary part.

## Author

L. M. Thompson

## Date

2019

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

- [src/mqc\\_algebra.F03](#)

## 6.10 mqc\_algebra::conjg Interface Reference

Returns the complex conjugate

### 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 Detailed Description

Returns the complex conjugate

#### 6.10.2 Member Function/Subroutine Documentation

##### 6.10.2.1 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

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

## 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

Contracts two arrays

### Public Member Functions

- type([mqc\\_scalar](#)) function [mqc\\_matrix\\_matrix\\_contraction](#) (Matrix1, Matrix2)

### 6.11.1 Detailed Description

Contracts two arrays

### 6.11.2 Member Function/Subroutine Documentation

#### 6.11.2.1 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

Returns the cosine

### 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 Detailed Description

Returns the cosine

## 6.13.2 Member Function/Subroutine Documentation

### 6.13.2.1 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	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](#)

## 6.14 mqc\_algebra::dagger Interface Reference

Returns the Hermitian conjugate

### Public Member Functions

- type(mqc\_vector) function [mqc\\_vector\\_conjugate\\_transpose](#) (Vector)  
*MQC\_Vector\_Conjugate\_Transpose is a function that returns the conjugate transpose of an MQC vector*
- type(mqc\_matrix) function [mqc\\_matrix\\_conjugate\\_transpose](#) (Matrix)  
*MQC\_Matrix\_Conjugate\_Transpose is a function that returns the conjugate transpose of a MQC matrix*

### 6.14.1 Detailed Description

Returns the Hermitian conjugate

### 6.14.2 Member Function/Subroutine Documentation

#### 6.14.2.1 `mqc_matrix_conjugate_transpose()`

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

**MQC\_Matrix\_Conjugate\_Transpose** is a function that returns the conjugate transpose of a MQC matrix

**Purpose:**

MQC\_Matrix\_Conjugate\_Transpose is a function that returns the conjugate transpose of a MQC matrix.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be conjugate transposed.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

#### 6.14.2.2 `mqc_vector_conjugate_transpose()`

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

**MQC\_Vector\_Conjugate\_Transpose** is a function that returns the conjugate transpose of an MQC vector

**Purpose:**

MQC\_Vector\_Conjugate\_Transpose is a function that returns the conjugate transpose of an MQC vector.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC vector to conjugate transpose.
----	---------------	--

## Author

L. M. Thompson

## Date

2017

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

Returns the dot product

## Public Member Functions

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

***MQC\_VectorVectorDotProduct** is a function that returns the dot product of two MQC vectors*

### 6.16.1 Detailed Description

Returns the dot product

### 6.16.2 Member Function/Subroutine Documentation

#### 6.16.2.1 `mqc_vectorvectordotproduct()`

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

**MQC\_VectorVectorDotProduct** is a function that returns the dot product of two MQC vectors

**Purpose:**

`MQC_VectorVectorDotProduct` is a function that returns the dot product of two MQC vectors. The first vector should be a row vector, while the second vector should be a column vector. The vectors should be of the same length.

**Parameters**

in	<i>Vector1</i>	Vector1 is Type(MQC_Vector) The MQC row vector.
in	<i>Vector2</i>	Vector2 is Type(MQC_Vector) The MQC column vector.

**Author**

H. P. Hratchian

L. M. Thompson



Date

2016

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

**Multiplies two arrays**

### 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 Detailed Description

**Multiplies two arrays**

## 6.18.2 Member Function/Subroutine Documentation

### 6.18.2.1 `mqc_matrixmatrixdotproduct()`

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

### 6.18.2.2 `mqc_matrixvectordotproduct()`

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

### 6.18.2.3 `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

Sets a symmetric packed intrinsic array as a square packed intrinsic array

### Public Member Functions

- subroutine [matrix\\_symm2sq\\_integer](#) (N, I\_Symm, I\_Sq)  
*Matrix\_Symm2Sq\_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array*
- subroutine [matrix\\_symm2sq\\_real](#) (N, A\_Symm, A\_Sq)  
*Matrix\_Symm2Sq\_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array*
- subroutine [matrix\\_symm2sq\\_complex](#) (N, A\_Symm, A\_Sq)  
*Matrix\_Symm2Sq\_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array*

### 6.20.1 Detailed Description

Sets a symmetric packed intrinsic array as a square packed intrinsic array

### 6.20.2 Member Function/Subroutine Documentation

#### 6.20.2.1 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 )
```

**Matrix\_Symm2Sq\_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array**

Purpose:

```
Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric-packed
intrinsic complex matrix to a rank-2 complex array.
TODO: Move this routine to MQC general
```

## Parameters

in	<i>N</i>	<i>N</i> is Integer(kind=int64) The leading dimension of symmetric-packed matrix <i>I_Symm</i> . unpacked.
in	<i>A_Symm</i>	<i>A_Symm</i> is Complex(kind=real64),Dimension(:) The symmetric-packed intrinsic complex matrix to be unpacked.
out	<i>A_Sq</i>	<i>A_Sq</i> is Complex(kind=real64),Dimension(N,N) The unpacked intrinsic complex matrix output.

## Author

L. M. Thompson

## Date

2017

## 6.20.2.2 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 )
```

**Matrix\_Symm2Sq\_Integer** is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array

## Purpose:

*Matrix\_Symm2Sq\_Integer* is a subroutine that converts a symmetric-packed intrinsic integer matrix to a rank-2 integer array.  
TODO: Move this routine to MQC general

## Parameters

in	<i>N</i>	<i>N</i> is Integer(kind=int64) The leading dimension of symmetric-packed matrix <i>I_Symm</i> . unpacked.
in	<i>I_Symm</i>	
Generated by Doxygen		<i>I_Symm</i> is Integer(kind=int64),Dimension(:) The symmetric-packed intrinsic integer matrix to be unpacked.
out	<i>I_Sq</i>	<i>I_Sq</i> is Integer(kind=int64),Dimension(N,N)

**Author**

H. P. Hratchian

**Date**

2017

**6.20.2.3 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 )

```

**Matrix\_Symm2Sq\_Real** is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array

**Purpose:**

Matrix\_Symm2Sq\_Real is a subroutine that converts a symmetric-packed intrinsic real matrix to a rank-2 real array.  
 TODO: Move this routine to MQC general

**Parameters**

in	<i>N</i>	N is Integer(kind=int64) The leading dimension of symmetric-packed matrix I_Symm. unpacked.
in	<i>A_Symm</i>	A_Symm is Real(kind=real64),Dimension(:) The symmetric-packed intrinsic real matrix to be unpacked.
out	<i>A_Sq</i>	A_Sq is Real(kind=real64),Dimension(N,N) The unpacked intrinsic real matrix output.

**Author**

H. P. Hratchian

Date

2017

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

- src/[mqc\\_algebra.F03](#)

## 6.21 mqc\_algebra::mqc\_cast\_complex Interface Reference

Sets an array to complex type

### Public Member Functions

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_complex](#) (VA)  
*MQC\_vector\_cast\_complex is a function that converts an MQC vector to its complex space*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_complex](#) (MA)  
*MQC\_Matrix\_Cast\_Complex is a function that converts an MQC matrix to its complex space*

### 6.21.1 Detailed Description

Sets an array to complex type

### 6.21.2 Member Function/Subroutine Documentation

#### 6.21.2.1 mqc\_matrix\_cast\_complex()

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

**MQC\_Matrix\_Cast\_Complex** is a function that converts an MQC matrix to its complex space

Purpose:

MQC\_Matrix\_Cast\_Complex is a function that converts an MQC matrix to its complex space.

**Parameters**

in	MA	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	----	--

**Author**

L. M. Thompson

**Date**

2017

**6.21.2.2 mqc\_vector\_cast\_complex()**

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

**MQC\_vector\_cast\_complex** is a function that converts an MQC vector to its complex space

**Purpose:**

MQC\_vector\_cast\_complex is a function that converts an MQC vector to its complex space.

**Parameters**

in	VA	VA is Class(MQC_Vector) The MQC vector to convert.
----	----	---

**Author**

L. M. Thompson

**Date**

2017

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

- src/[mqc\\_algebra.F03](#)



## 6.22 mqc\_algebra::mqc\_cast\_integer Interface Reference

Sets an array to integer type

### Public Member Functions

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_integer](#) (VA)  
*MQC\_vector\_cast\_integer is a function that converts an MQC vector to its integer space*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_integer](#) (MA)  
*MQC\_Matrix\_Cast\_Integer is a function that converts an MQC matrix to its integer space*

### 6.22.1 Detailed Description

Sets an array to integer type

### 6.22.2 Member Function/Subroutine Documentation

#### 6.22.2.1 mqc\_matrix\_cast\_integer()

```
type(mqc\_matrix) function mqc_algebra::mqc_cast_integer::mqc_matrix_cast_integer (
    type(mqc\_matrix), intent(in) MA )
```

**MQC\_Matrix\_Cast\_Integer is a function that converts an MQC matrix to its integer space**

**Purpose:**

MQC\_Matrix\_Cast\_Integer is a function that converts an MQC matrix to its integer space.

**Parameters**

in	<i>MA</i>	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	-----------	--

**Author**

L. M. Thompson

Date

2019

### 6.22.2.2 mqc\_vector\_cast\_integer()

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

**MQC\_vector\_cast\_integer** is a function that converts an MQC vector to its integer space

Purpose:

MQC\_vector\_cast\_integer is a function that converts an MQC vector to its integer space.

Parameters

in	VA	VA is Type(MQC_Vector) The MQC vector to convert.
----	----	--

Author

L. M. Thompson

Date

2019

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

- [src/mqc\\_algebra.F03](#)

## 6.23 mqc\_algebra::mqc\_cast\_real Interface Reference

Sets an array to real type

### Public Member Functions

- type([mqc\\_vector](#)) function [mqc\\_vector\\_cast\\_real](#) (VA)  
*MQC\_vector\_cast\_real is a function that converts an MQC vector to its real space*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_cast\\_real](#) (MA)  
*MQC\_Matrix\_Cast\_Real is a function that converts an MQC matrix to its real space*

### 6.23.1 Detailed Description

Sets an array to real type

### 6.23.2 Member Function/Subroutine Documentation

#### 6.23.2.1 mqc\_matrix\_cast\_real()

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

**MQC\_Matrix\_Cast\_Real** is a function that converts an MQC matrix to its real space

**Purpose:**

MQC\_Matrix\_Cast\_Real is a function that converts an MQC matrix to its real space.

**Parameters**

in	MA	MA is Type(MQC_Matrix) The MQC matrix to convert.
----	----	--

**Author**

X. Sheng

**Date**

2017

#### 6.23.2.2 mqc\_vector\_cast\_real()

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

**MQC\_vector\_cast\_real** is a function that converts an MQC vector to its real space

**Purpose:**

MQC\_vector\_cast\_real is a function that converts an MQC vector to its real space.

**Parameters**

in	VA	VA is Class(MQC_Vector) The MQC vector to convert.
----	----	---

**Author**

X. Sheng

**Date**

2017

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

- [src/mqc\\_algebra.F03](#)

## 6.24 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](#)

### 6.24.1 Member Data Documentation

#### 6.24.1.1 nalpstr

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

#### 6.24.1.2 nbetstr

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

### 6.24.1.3 ndets

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

### 6.24.1.4 order

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

### 6.24.1.5 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](#)

## 6.25 mqc\_est::mqc\_determinant\_string Type Reference

### Public Attributes

- `type(mqc_matrix)` [alpha](#)
- `type(mqc_matrix)` [beta](#)

### 6.25.1 Member Data Documentation

#### 6.25.1.1 alpha

```
type(mqc_matrix) mqc_est::mqc_determinant_string::alpha
```

#### 6.25.1.2 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](#)

## 6.26 mqc\_algebra::mqc\_have\_complex Interface Reference

Determines in an array is complex type

### Public Member Functions

- logical function [mqc\\_vector\\_havecomplex](#) (Vector)  
*MQC\_Vector\_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector*
- logical function [mqc\\_matrix\\_havecomplex](#) (Matrix)  
*MQC\_Matrix\_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix*

### 6.26.1 Detailed Description

Determines in an array is complex type

### 6.26.2 Member Function/Subroutine Documentation

#### 6.26.2.1 mqc\_matrix\_havecomplex()

```
logical function mqc_algebra::mqc_have_complex::mqc_matrix_havecomplex (
    type(mqc\_matrix), intent(in) Matrix )
```

**MQC\_Matrix\_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix**

**Purpose:**

MQC\_Matrix\_HaveComplex is a function that returns TRUE if an MQC matrix has an allocated complex matrix and FALSE if it does not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

Date

2017

### 6.26.2.2 mqc\_vector\_havecomplex()

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

**MQC\_Vector\_HaveComplex** is a function that returns **TRUE** or **FALSE** indicating whether the MQC vector has an allocated complex vector

Purpose:

MQC\_Vector\_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector.

Parameters

in, out	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	---------------	--

Author

L. M. Thompson

Date

2017

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

- src/[mqc\\_algebra.F03](#)

## 6.27 mqc\_algebra::mqc\_have\_int Interface Reference

**Determines in an array is integer type**

## Public Member Functions

- logical function [mqc\\_vector\\_haveinteger](#) (Vector)  
*MQC\_Vector\_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector*
- logical function [mqc\\_matrix\\_haveinteger](#) (Matrix)  
*MQC\_Matrix\_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix*

### 6.27.1 Detailed Description

Determines in an array is integer type

### 6.27.2 Member Function/Subroutine Documentation

#### 6.27.2.1 mqc\_matrix\_haveinteger()

```
logical function mqc_algebra::mqc_have_int::mqc_matrix_haveinteger (
    type(mqc\_matrix), intent(in) Matrix )
```

**MQC\_Matrix\_HaveInteger** is a function used to indicate if an MQC matrix has an allocated integer matrix

Purpose:

MQC\_Matrix\_HaveInteger is a function that returns TRUE if an MQC matrix has an allocated integer matrix and FALSE if it does not.

Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

Author

L. M. Thompson

Date

2016



### 6.27.2.2 mqc\_vector\_haveinteger()

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

**MQC\_Vector\_HaveInteger** is a function that returns **TRUE** or **FALSE** indicating whether the MQC vector has an allocated integer vector

#### Purpose:

MQC\_Vector\_HaveInteger is a function that returns **TRUE** or **FALSE** indicating whether the MQC vector has an allocated integer vector.

#### Parameters

in, out	Vector	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
---------	--------	--

#### Author

H. P. Hratchian

#### Date

2016

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

- src/[mqc\\_algebra.F03](#)

## 6.28 mqc\_algebra::mqc\_have\_real Interface Reference

Determines in an array is real type

### Public Member Functions

- logical function [mqc\\_vector\\_havereal](#) (Vector)  
*MQC\_Vector\_HaveReal is a function that returns **TRUE** or **FALSE** indicating whether the MQC vector has an allocated real vector*
- logical function [mqc\\_matrix\\_havereal](#) (Matrix)  
*MQC\_Matrix\_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix*

### 6.28.1 Detailed Description

Determines in an array is real type

### 6.28.2 Member Function/Subroutine Documentation

#### 6.28.2.1 mqc\_matrix\_havereal()

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

**MQC\_Matrix\_HaveReal** is a function used to indicate if an MQC matrix has an allocated real matrix

**Purpose:**

MQC\_Matrix\_HaveReal is a function that returns TRUE if an MQC matrix has an allocated real matrix and FALSE if it does not.

**Parameters**

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.
----	---------------	--

**Author**

L. M. Thompson

**Date**

2016

#### 6.28.2.2 mqc\_vector\_havereal()

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

**MQC\_Vector\_HaveReal** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector

**Purpose:**

MQC\_Vector\_HaveReal is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector.

## Parameters

in	Vector	Vector is Type(MQC_Vector) The MQC_Vector to be tested.
----	--------	--

## Author

H. P. Hratchian

## Date

2016

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

- [src/mqc\\_algebra.F03](#)

## 6.29 mqc\_algebra::mqc\_matrix Type Reference

### Rank 2 array variable

### Public Member Functions

- Procedure, public [print](#) => [mqc\\_print\\_matrix\\_algebra1](#)  
*Print the MQC Matrix*
- Procedure, public [init](#) => [mqc\\_matrix\\_initialize](#)  
*Initialize the MQC Matrix with a specified value*
- Procedure, public [identity](#) => [mqc\\_matrix\\_identity](#)  
*Initialize the MQC Matrix as the identity matrix*
- Procedure, public [set](#) => [mqc\\_matrix\\_set](#)  
*Set all elements in the MQC Matrix to a specified value*
- Procedure, public [norm](#) => [mqc\\_matrix\\_norm](#)  
*Returns the norm of the MQC Matrix*
- Procedure, public [transpose](#) => [mqc\\_matrix\\_transpose](#)  
*Returns the transpose of the MQC Matrix*
- Procedure, public [dagger](#) => [mqc\\_matrix\\_conjugate\\_transpose](#)  
*Returns the Hermitian transpose of the MQC Matrix*
- Procedure, public [diag](#) => [mqc\\_matrix\\_diagonalize](#)  
*Returns the eigenvalues and eigenvalues of a symmetric or hermitian MQC Matrix*
- Procedure, public [svd](#) => [mqc\\_matrix\\_svd](#)  
*Computes the singular value decomposition of the MQC Matrix*
- Procedure, public [eigensys](#) => [mqc\\_matrix\\_generalized\\_eigensystem](#)

*Solves the generalized eigenproblem of the MQC Matrix*

- Procedure, public `inv` => `mqc_matrix_inverse`

*Returns the inverse of the MQC Matrix*

- Procedure, public `det` => `mqc_matrix_determinant`

*Returns the determinant of the MQC Matrix*

- Procedure, public `trace` => `mqc_matrix_trace`

*Returns the trace of the MQC Matrix*

- Procedure, public `rmsmax` => `mqc_matrix_rms_max`

*Returns the root mean square deviation and maximum deviation of elements of the MQC Matrix*

- Procedure, public `sqrt` => `mqc_matrix_sqrt`

*Returns the square root of the MQC Matrix elements*

- Procedure, public `at` => `mqc_matrix_scalar_at`

*Returns the value of the specified element of the MQC Matrix*

- Procedure, public `vat` => `mqc_matrix_vector_at`

*Returns the vector of the specified subvector of the MQC Matrix*

- Procedure, public `mat` => `mqc_matrix_matrix_at`

*Returns the matrix of the specified submatrix of the MQC Matrix*

- Procedure, public `put` => `mqc_matrix_scalar_put`

*Updates the specified element of the MQC Matrix to the specified value*

- Procedure, public `vput` => `mqc_matrix_vector_put`

*Updates the specified subvector of the MQC Matrix to the specified vector*

- Procedure, public `mput` => `mqc_matrix_matrix_put`

*Updates the specified submatrix of the MQC Matrix to the specified matrix*

## 6.29.1 Detailed Description

Rank 2 array variable

## 6.29.2 Member Function/Subroutine Documentation

### 6.29.2.1 `at()`

```
Procedure, public mqc_algebra::mqc_matrix::at ( )
```

Returns the value of the specified element of the MQC Matrix

### 6.29.2.2 `dagger()`

```
Procedure, public mqc_algebra::mqc_matrix::dagger ( )
```

Returns the Hermitian transpose of the MQC Matrix

### 6.29.2.3 det()

Procedure, public mqc\_algebra::mqc\_matrix::det ( )

**Returns the determinant of the MQC Matrix**

### 6.29.2.4 diag()

Procedure, public mqc\_algebra::mqc\_matrix::diag ( )

**Returns the eigenvalues and eigenvalues of a symmetric or hermitian MQC Matrix**

### 6.29.2.5 eigensys()

Procedure, public mqc\_algebra::mqc\_matrix::eigensys ( )

**Solves the generalized eigenproblem of the MQC Matrix**

### 6.29.2.6 identity()

Procedure, public mqc\_algebra::mqc\_matrix::identity ( )

**Initialize the MQC Matrix as the identity matrix**

### 6.29.2.7 init()

Procedure, public mqc\_algebra::mqc\_matrix::init ( )

**Initialize the MQC Matrix with a specified value**

### 6.29.2.8 inv()

Procedure, public mqc\_algebra::mqc\_matrix::inv ( )

**Returns the inverse of the MQC Matrix**

**6.29.2.9 mat()**

```
Procedure, public mqc_algebra::mqc_matrix::mat ( )
```

**Returns the matrix of the specified submatrix of the MQC Matrix**

**6.29.2.10 mput()**

```
Procedure, public mqc_algebra::mqc_matrix::mput ( )
```

**Updates the specified submatrix of the MQC Matrix to the specified matrix**

**6.29.2.11 norm()**

```
Procedure, public mqc_algebra::mqc_matrix::norm ( )
```

**Returns the norm of the MQC Matrix**

**6.29.2.12 print()**

```
Procedure, public mqc_algebra::mqc_matrix::print ( )
```

**Print the MQC Matrix**

**6.29.2.13 put()**

```
Procedure, public mqc_algebra::mqc_matrix::put ( )
```

**Updates the specified element of the MQC Matrix to the specified value**

**6.29.2.14 rmsmax()**

```
Procedure, public mqc_algebra::mqc_matrix::rmsmax ( )
```

**Returns the root mean square deviation and maximum deviation of elements of the MQC Matrix**

**6.29.2.15 set()**

```
Procedure, public mqc_algebra::mqc_matrix::set ( )
```

**Set all elements in the MQC Matrix to a specified value**

**6.29.2.16 sqrt()**

```
Procedure, public mqc_algebra::mqc_matrix::sqrt ( )
```

**Returns the square root of the MQC Matrix elements**

**6.29.2.17 svd()**

```
Procedure, public mqc_algebra::mqc_matrix::svd ( )
```

**Computes the singular value decomposition of the MQC Matrix**

**6.29.2.18 trace()**

```
Procedure, public mqc_algebra::mqc_matrix::trace ( )
```

**Returns the trace of the MQC Matrix**

**6.29.2.19 transpose()**

```
Procedure, public mqc_algebra::mqc_matrix::transpose ( )
```

**Returns the transpose of the MQC Matrix**

**6.29.2.20 vat()**

```
Procedure, public mqc_algebra::mqc_matrix::vat ( )
```

**Returns the vector of the specified subvector of the MQC Matrix**

### 6.29.2.21 vput()

Procedure, public `mqc_algebra::mqc_matrix::vput ( )`

**Updates the specified subvector of the MQC Matrix to the specified vector**

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

- [src/mqc\\_algebra.F03](#)

## 6.30 mqc\_algebra::mqc\_matrix\_diagmatrix\_put Interface Reference

**Sets a diagonal packed intrinsic array as an MQC Matrix object**

### Public Member Functions

- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_integer](#) (mat, diagMatrixIn)  
*MQC\_Matrix\_DiagMatrix\_Put\_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_real](#) (mat, diagMatrixIn)  
*MQC\_Matrix\_DiagMatrix\_Put\_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_complex](#) (mat, diagMatrixIn)  
*MQC\_Matrix\_DiagMatrix\_Put\_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector*
- subroutine [mqc\\_matrix\\_diagmatrix\\_put\\_vector](#) (diagVectorIn, mat)  
*MQC\_Matrix\_DiagMatrix\_Put\_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector*

### 6.30.1 Detailed Description

**Sets a diagonal packed intrinsic array as an MQC Matrix object**

### 6.30.2 Member Function/Subroutine Documentation

#### 6.30.2.1 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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector**

**Purpose:**

`MQC_Matrix_DiagMatrix_Put_Complex` is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector.



## Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>Diag↔ MatrixIn</i>	DiagMatrixIn is complex(kind=real64),dimension(:) Intrinsic complex vector to write as diagonal matrix.

## Author

L. M. Thompson

## Date

2017

## 6.30.2.2 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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_integer** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector

## Purpose:

MQC\_Matrix\_DiagMatrix\_Put\_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector.

## Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>Diag↔ MatrixIn</i>	DiagMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as diagonal matrix.

**Author**

H. P. Hratchian

**Date**

2017

**6.30.2.3 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 )

```

**MQC\_Matrix\_DiagMatrix\_Put\_Real** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector

**Purpose:**

MQC\_Matrix\_DiagMatrix\_Put\_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector.

**Parameters**

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	<i>Diag↔ MatrixIn</i>	DiagMatrixIn is real(kind=real64),dimension(:) Intrinsic real vector to write as diagonal matrix.

**Author**

H. P. Hratchian

**Date**

2017

### 6.30.2.4 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 )
```

**MQC\_Matrix\_DiagMatrix\_Put\_Vector** is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

#### Purpose:

MQC\_Matrix\_DiagMatrix\_Put\_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector.

#### Parameters

in	<i>Diag↔ VectorIn</i>	DiagVectorIn is class(MQC_Vector) Name of the MQC vector to write as diagonal matrix.
in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.

#### Author

L. M. Thompson

#### Date

2018

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

- src/[mqc\\_algebra.F03](#)

## 6.31 mqc\_algebra::mqc\_matrix\_symmmatrix\_put Interface Reference

Sets a symmetric packed intrinsic array as an MQC Matrix object

### Public Member Functions

- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_integer](#) (mat, symmMatrixIn)  
*MQC\_Matrix\_SymmMatrix\_Put\_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_real](#) (mat, symmMatrixIn)  
*MQC\_Matrix\_SymmMatrix\_Put\_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_matrix\\_symmmatrix\\_put\\_complex](#) (mat, symmMatrixIn)  
*MQC\_Matrix\_SymmMatrix\_Put\_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector*

### 6.31.1 Detailed Description

Sets a symmetric packed intrinsic array as an MQC Matrix object

### 6.31.2 Member Function/Subroutine Documentation

#### 6.31.2.1 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Complex** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector

#### Purpose:

MQC\_Matrix\_SymmMatrix\_Put\_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector.

#### Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	<i>SymmMatrixIn</i>	SymmMatrixIn is complex(kind=real64),dimension(:) Intrinsic complex vector to write as symmetric-packed matrix.

#### Author

L. M. Thompson

#### Date

2017

### 6.31.2.2 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Integer** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector

#### Purpose:

MQC\_Matrix\_SymmMatrix\_Put\_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector.

#### Parameters

in, out	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	<i>SymmMatrixIn</i>	SymmMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as symmetric-packed matrix.

#### Author

H. P. Hratchian

#### Date

2017

### 6.31.2.3 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 )
```

**MQC\_Matrix\_SymmMatrix\_Put\_Real** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector

#### Purpose:

MQC\_Matrix\_SymmMatrix\_Put\_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector.

## Parameters

<code>in, out</code>	<i>mat</i>	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
<code>in</code>	<i>Symm↔ MatrixIn</i>	SymmMatrixIn is <code>real(kind=real64), dimension(:)</code> Intrinsic real vector to write as symmetric-packed matrix.

## Author

H. P. Hratchian

## Date

2017

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

- [src/mqc\\_algebra.F03](#)

## 6.32 mqc\_est::mqc\_matrix\_undospinblockghf Interface Reference

### Public Member Functions

- subroutine [mqc\\_matrix\\_undospinblockghf\\_eigenvalues](#) (eigenvaluesIn, vectorOut)
- subroutine [mqc\\_matrix\\_undospinblockghf\\_integral](#) (integralIn, matrixOut)

### 6.32.1 Member Function/Subroutine Documentation

#### 6.32.1.1 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 )
```

### 6.32.1.2 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](#)

## 6.33 mqc\_algebra::mqc\_print Interface Reference

Prints an object

### 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)  
***MQC\_Print\_Vector\_Algebra1 is a subroutine used to print an MQC vector***
- subroutine [mqc\\_print\\_matrix\\_algebra1](#) (Matrix, IOut, Header, Blank\_At\_Top, Blank\_At\_Bottom)  
***MQC\_Print\_Matrix\_Algebra1 is a subroutine used to print an MQC matrix***
- subroutine [mqc\\_print\\_r4tensor\\_algebra1](#) (Tensor, IOut, Header, blank\_at\_top, blank\_at\_bottom)

### 6.33.1 Detailed Description

Prints an object

### 6.33.2 Member Function/Subroutine Documentation

#### 6.33.2.1 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 )
```

**MQC\_Print\_Matrix\_Algebra1 is a subroutine used to print an MQC matrix**

Purpose:

MQC\_Print\_Matrix\_Algebra1 is a subroutine used to print an MQC matrix.  
Blank\_At\_Top and Blank\_At\_Bottom are optional logical arguments to print blank lines before or after output.

## Parameters

in	<i>Matrix</i>	Matrix is Class(MQC_Matrix) 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 Matrix.
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

## 6.33.2.2 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 )

```



### 6.33.2.3 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.
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

### 6.33.2.4 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 )
```

**MQC\_Print\_Vector\_Algebra1** is a subroutine used to print an MQC vector

#### Purpose:

MQC\_Print\_Vector\_Algebra1 is a subroutine used to print an MQC vector.  
Blank\_At\_Top and Blank\_At\_Bottom are optional logical arguments to print blank lines before or after output.

#### Parameters

in	<i>Vector</i>	Vector is Class(MQC_Vector) 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 Vector.
in	<i>Verbose</i>	Verbose is Logical,Optional Adds extra printing to output.
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

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

- [src/mqc\\_algebra.F03](#)

## 6.34 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)

### 6.34.1 Member Function/Subroutine Documentation

#### 6.34.1.1 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 )
```

#### 6.34.1.2 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 )
```

### 6.34.1.3 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 )
```

### 6.34.1.4 mqc\_print\_wavefunction()

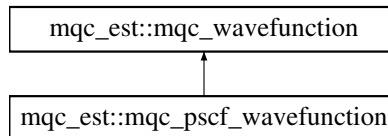
```
subroutine mqc_est::mqc_print::mqc_print_wavefunction (
    class(mqc_wavefunction) wavefunction,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in), optional label )
```

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

- src/[mqc\\_est.F03](#)

## 6.35 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

#### 6.35.1 Member Data Documentation

### 6.35.1.1 nactive

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nactive
```

### 6.35.1.2 ncore

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::ncore
```

### 6.35.1.3 nfrz

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nfrz
```

### 6.35.1.4 nval

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nval
```

### 6.35.1.5 pscf\_amplitudes

```
type(mqc_matrix) mqc_est::mqc_pscf_wavefunction::pscf_amplitudes
```

### 6.35.1.6 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](#)

## 6.36 mqc\_algebra::mqc\_r4tensor Type Reference

Updates the specified element of the MQC Matrix to the specified value

## Public Member Functions

- Procedure, public `print` => `mqc_print_r4tensor_algebra1`  
*Print the MQC R4Tensor*
- Procedure, public `at` => `mqc_r4tensor_at`  
*Return the specified element in the MQC R4Tensor*
- Procedure, public `put` => `mqc_r4tensor_put`  
*Update the specified element in the MQC R4Tensor with the specified value*
- Procedure, public `init` => `mqc_r4tensor_initialize`  
*Initialize the MQC R4Tensor*

### 6.36.1 Detailed Description

Updates the specified element of the MQC Matrix to the specified value

Rank 4 array variable

### 6.36.2 Member Function/Subroutine Documentation

#### 6.36.2.1 `at()`

```
Procedure, public mqc_algebra::mqc_r4tensor::at ( )
```

**Return the specified element in the MQC R4Tensor**

#### 6.36.2.2 `init()`

```
Procedure, public mqc_algebra::mqc_r4tensor::init ( )
```

**Initialize the MQC R4Tensor**

#### 6.36.2.3 `print()`

```
Procedure, public mqc_algebra::mqc_r4tensor::print ( )
```

**Print the MQC R4Tensor**

### 6.36.2.4 put()

Procedure, public mqc\_algebra::mqc\_r4tensor::put ( )

**Update the specified element in the MQC R4Tensor with the specified value**

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

- src/[mqc\\_algebra.F03](#)

## 6.37 mqc\_algebra::mqc\_scalar Type Reference

Rank 0 array variable

### Public Member Functions

- Procedure, public [print](#) => [mqc\\_print\\_scalar\\_algebra1](#)  
*Print the MQC Scalar*
- Procedure, public [rval](#) => [mqc\\_scalar\\_get\\_intrinsic\\_real](#)  
*Return the value of MQC Scalar as an intrinsic real*
- Procedure, public [ival](#) => [mqc\\_scalar\\_get\\_intrinsic\\_integer](#)  
*Return the value of MQC Scalar as an intrinsic integer*
- Procedure, public [cval](#) => [mqc\\_scalar\\_get\\_intrinsic\\_complex](#)  
*Return the value of MQC Scalar as an intrinsic complex*
- Procedure, public [abs](#) => [mqc\\_scalar\\_get\\_abs\\_value](#)  
*Take the absolute value of the MQC Scalar*
- Procedure, public [random](#) => [mqc\\_scalar\\_get\\_random\\_value](#)  
*Return a random value to the MQC Scalar*

### 6.37.1 Detailed Description

Rank 0 array variable

### 6.37.2 Member Function/Subroutine Documentation

#### 6.37.2.1 abs()

Procedure, public mqc\_algebra::mqc\_scalar::abs ( )

**Take the absolute value of the MQC Scalar**

### 6.37.2.2 cval()

Procedure, public mqc\_algebra::mqc\_scalar::cval ( )

**Return the value of MQC Scalar as an intrinsic complex**

### 6.37.2.3 ival()

Procedure, public mqc\_algebra::mqc\_scalar::ival ( )

**Return the value of MQC Scalar as an intrinsic integer**

### 6.37.2.4 print()

Procedure, public mqc\_algebra::mqc\_scalar::print ( )

**Print the MQC Scalar**

### 6.37.2.5 random()

Procedure, public mqc\_algebra::mqc\_scalar::random ( )

**Return a random value to the MQC Scalar**

### 6.37.2.6 rval()

Procedure, public mqc\_algebra::mqc\_scalar::rval ( )

**Return the value of MQC Scalar as an intrinsic real**

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

- [src/mqc\\_algebra.F03](#)



## 6.38 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](#)

### 6.38.1 Member Function/Subroutine Documentation

#### 6.38.1.1 addlabel()

Procedure, public mqc\_est::mqc\_scf\_eigenvalues::addlabel ( )

#### 6.38.1.2 at()

Procedure, public mqc\_est::mqc\_scf\_eigenvalues::at ( )

#### 6.38.1.3 getblock()

Procedure, public mqc\_est::mqc\_scf\_eigenvalues::getblock ( )

#### 6.38.1.4 getlabel()

Procedure, public mqc\_est::mqc\_scf\_eigenvalues::getlabel ( )

#### 6.38.1.5 power()

Procedure, public mqc\_est::mqc\_scf\_eigenvalues::power ( )

### 6.38.1.6 print()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::print ( )
```

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

- src/[mqc\\_est.F03](#)

## 6.39 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](#)

### 6.39.1 Member Function/Subroutine Documentation

#### 6.39.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_integral::addlabel ( )
```

#### 6.39.1.2 deleteelist()

```
Procedure, public mqc_est::mqc_scf_integral::deleteelist ( )
```

**6.39.1.3 det()**

Procedure, public mqc\_est::mqc\_scf\_integral::det ( )

**6.39.1.4 diag()**

Procedure, public mqc\_est::mqc\_scf\_integral::diag ( )

**6.39.1.5 eigensys()**

Procedure, public mqc\_est::mqc\_scf\_integral::eigensys ( )

**6.39.1.6 getblock()**

Procedure, public mqc\_est::mqc\_scf\_integral::getblock ( )

**6.39.1.7 getelist()**

Procedure, public mqc\_est::mqc\_scf\_integral::getelist ( )

**6.39.1.8 getlabel()**

Procedure, public mqc\_est::mqc\_scf\_integral::getlabel ( )

**6.39.1.9 identity()**

Procedure, public mqc\_est::mqc\_scf\_integral::identity ( )

**6.39.1.10 init()**

Procedure, public mqc\_est::mqc\_scf\_integral::init ( )

**6.39.1.11 inv()**

Procedure, public mqc\_est::mqc\_scf\_integral::inv ( )

**6.39.1.12 norm()**

Procedure, public mqc\_est::mqc\_scf\_integral::norm ( )

**6.39.1.13 orbitals()**

Procedure, public mqc\_est::mqc\_scf\_integral::orbitals ( )

**6.39.1.14 print()**

Procedure, public mqc\_est::mqc\_scf\_integral::print ( )

**6.39.1.15 setelist()**

Procedure, public mqc\_est::mqc\_scf\_integral::setelist ( )

**6.39.1.16 swap()**

Procedure, public mqc\_est::mqc\_scf\_integral::swap ( )

### 6.39.1.17 trace()

Procedure, public mqc\_est::mqc\_scf\_integral::trace ( )

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

- src/[mqc\\_est.F03](#)

## 6.40 mqc\_algebra::mqc\_set\_array2vector Interface Reference

Sets an intrinsic array as an MQC Algebra object

### Public Member Functions

- subroutine [mqc\\_set\\_array2vector\\_integer](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector*
- subroutine [mqc\\_set\\_array2vector\\_real](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector*
- subroutine [mqc\\_set\\_array2vector\\_complex](#) (VectorOut, ArrayIn)  
*MQC\_Set\_Array2Vector\_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector*

### 6.40.1 Detailed Description

Sets an intrinsic array as an MQC Algebra object

### 6.40.2 Member Function/Subroutine Documentation

#### 6.40.2.1 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 )
```

**MQC\_Set\_Array2Vector\_Complex** is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

Purpose:

MQC\_Set\_Array2Vector\_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector.

## Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Complex(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

## Author

L. M. Thompson

## Date

2017

## 6.40.2.2 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 )
```

**MQC\_Set\_Array2Vector\_Integer** is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

## Purpose:

MQC\_Set\_Array2Vector\_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

## Parameters

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Integer(kind=int64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

H. P. Hratchian

**Date**

2016

**6.40.2.3 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 )
```

**MQC\_Set\_Array2Vector\_Real** is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

**Purpose:**

MQC\_Set\_Array2Vector\_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector.

**Parameters**

in, out	<i>VectorOut</i>	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	<i>ArrayIn</i>	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

**Author**

H. P. Hratchian

**Date**

2016

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

- src/[mqc\\_algebra.F03](#)

## 6.41 mqc\_est::mqc\_twoeris Type Reference

### Public Member Functions

- procedure, public [print](#) => [mqc\\_print\\_twoeris](#)

### 6.41.1 Member Function/Subroutine Documentation

#### 6.41.1.1 print()

```
procedure, public mqc_est::mqc_twoeris::print ( )
```

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

- src/[mqc\\_est.F03](#)

## 6.42 mqc\_algebra::mqc\_vector Type Reference

Rank 1 array variable

### Public Member Functions

- Procedure, public [print](#) => [mqc\\_print\\_vector\\_algebra1](#)  
*Print the MQC Vector*
- Procedure, public [size](#) => [mqc\\_length\\_vector](#)  
*Returns the length of the MQC Vector*
- Procedure, public [init](#) => [mqc\\_vector\\_initialize](#)  
*Initilizes the MQC Vector*
- Procedure, public [norm](#) => [mqc\\_vector\\_norm](#)  
*Returns the norm of the MQC Vector*
- Procedure, public [transpose](#) => [mqc\\_vector\\_transpose](#)  
*Returns the transpose of the MQC Vector*
- Procedure, public [dagger](#) => [mqc\\_vector\\_conjugate\\_transpose](#)  
*Returns the Hermitian conjugate of the MQC Vector*
- Procedure, public [at](#) => [mqc\\_vector\\_scalar\\_at](#)  
*Returns the value at the specified element of the MQC Vector*
- Procedure, public [vat](#) => [mqc\\_vector\\_vector\\_at](#)  
*Returns the subvector between specified element of the MQC Vector*
- Procedure, public [put](#) => [mqc\\_vector\\_scalar\\_put](#)  
*Updates the specified element of the MQC\_Vector with the specified value*



- Procedure, public `vput` => `mqc_vector_vector_put`  
*Updates the specified subvector of the MQC\_Vector with the specified vector*
- Procedure, public `push` => `mqc_vector_push`  
*Appends the specified value to the end of the MQC\_Vector*
- Procedure, public `unshift` => `mqc_vector_unshift`  
*Prepends the specified value to the beginning of the MQC\_Vector*
- Procedure, public `pop` => `mqc_vector_pop`  
*Removes the last element of the MQC\_Vector and returns the value*
- Procedure, public `shift` => `mqc_vector_shift`  
*Removes the first element of the MQC\_Vector and returns the value*
- Procedure, public `maxval` => `mqc_vector_maxval`  
*Returns the maximum value in the MQC\_Vector*
- Procedure, public `minval` => `mqc_vector_minloc`  
*Returns the minimum value in the MQC\_Vector*
- Procedure, public `maxloc` => `mqc_vector_maxval`  
*Returns the location of the maximum value in the MQC\_Vector*
- Procedure, public `minloc` => `mqc_vector_minloc`  
*Returns the location of the minimum value in the MQC\_Vector*
- Procedure, public `argsort` => `mqc_vector_argsort`  
*Returns the indices of the MQC\_Vector sorted from low to high*
- Procedure, public `sort` => `mqc_vector_sort`  
*Returns the MQC\_Vector sorted from low to high unless vector specifying index order is provided*
- Procedure, public `sqr` => `mqc_vector_sqr`  
*Returns the square root of each element in the MQC\_Vector*
- Procedure, public `abs` => `mqc_vector_abs`  
*Returns the absolute value of each element in the MQC\_Vector*
- Procedure, public `power` => `mqc_vector_power`  
*Returns each element in the MQC\_Vector raised to a specified power*
- Procedure, public `diag` => `mqc_matrix_diagmatrix_put_vector`, `mqc_vector2diagmatrix`  
*Returns a diagonal MQC Matrix with values specified by the MQC\_Vector*

## 6.42.1 Detailed Description

Rank 1 array variable

## 6.42.2 Member Function/Subroutine Documentation

### 6.42.2.1 `abs()`

Procedure, public `mqc_algebra::mqc_vector::abs` ( )

Returns the absolute value of each element in the MQC\_Vector

#### 6.42.2.2 argsort()

Procedure, public mqc\_algebra::mqc\_vector::argsort ( )

**Returns the indices of the MQC\_Vector sorted from low to high**

#### 6.42.2.3 at()

Procedure, public mqc\_algebra::mqc\_vector::at ( )

**Returns the value at the specified element of the MQC Vector**

#### 6.42.2.4 dagger()

Procedure, public mqc\_algebra::mqc\_vector::dagger ( )

**Returns the Hermitian conjugate of the MQC Vector**

#### 6.42.2.5 diag()

Procedure, public mqc\_algebra::mqc\_vector::diag ( )

**Returns a diagonal MQC Matrix with values specified by the MQC\_Vector**

#### 6.42.2.6 init()

Procedure, public mqc\_algebra::mqc\_vector::init ( )

**Initilizes the MQC Vector**

#### 6.42.2.7 maxloc()

Procedure, public mqc\_algebra::mqc\_vector::maxloc ( )

**Returns the location of the maximum value in the MQC\_Vector**

#### 6.42.2.8 maxval()

Procedure, public mqc\_algebra::mqc\_vector::maxval ( )

**Returns the maximum value in the MQC\_Vector**

#### 6.42.2.9 minloc()

Procedure, public mqc\_algebra::mqc\_vector::minloc ( )

**Returns the location of the minimum value in the MQC\_Vector**

#### 6.42.2.10 minval()

Procedure, public mqc\_algebra::mqc\_vector::minval ( )

**Returns the minimum value in the MQC\_Vector**

#### 6.42.2.11 norm()

Procedure, public mqc\_algebra::mqc\_vector::norm ( )

**Returns the norm of the MQC Vector**

#### 6.42.2.12 pop()

Procedure, public mqc\_algebra::mqc\_vector::pop ( )

**Removes the last element of the MQC\_Vector and returns the value**

#### 6.42.2.13 power()

Procedure, public mqc\_algebra::mqc\_vector::power ( )

**Returns each element in the MQC\_Vector raised to a specified power**

**6.42.2.14 print()**

```
Procedure, public mqc_algebra::mqc_vector::print ( )
```

**Print the MQC Vector**

**6.42.2.15 push()**

```
Procedure, public mqc_algebra::mqc_vector::push ( )
```

**Appends the specified value to the end of the MQC\_Vector**

**6.42.2.16 put()**

```
Procedure, public mqc_algebra::mqc_vector::put ( )
```

**Updates the specified element of the MQC\_Vector with the specified value**

**6.42.2.17 shift()**

```
Procedure, public mqc_algebra::mqc_vector::shift ( )
```

**Removes the first element of the MQC\_Vector and returns the value**

**6.42.2.18 size()**

```
Procedure, public mqc_algebra::mqc_vector::size ( )
```

**Returns the length of the MQC Vector**

**6.42.2.19 sort()**

```
Procedure, public mqc_algebra::mqc_vector::sort ( )
```

**Returns the MQC\_Vector sorted from low to high unless vector specifying index order is provided**

#### 6.42.2.20 sqrt()

Procedure, public mqc\_algebra::mqc\_vector::sqrt ( )

**Returns the square root of each element in the MQC\_Vector**

#### 6.42.2.21 transpose()

Procedure, public mqc\_algebra::mqc\_vector::transpose ( )

**Returns the transpose of the MQC Vector**

#### 6.42.2.22 unshift()

Procedure, public mqc\_algebra::mqc\_vector::unshift ( )

**Prepends the specified value to the beginning of the MQC\_Vector**

#### 6.42.2.23 vat()

Procedure, public mqc\_algebra::mqc\_vector::vat ( )

**Returns the subvector between specified element of the MQC Vector**

#### 6.42.2.24 vput()

Procedure, public mqc\_algebra::mqc\_vector::vput ( )

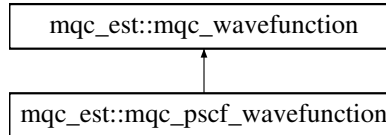
**Updates the specified subvector of the MQC\_Vector with the specified vector**

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

- [src/mqc\\_algebra.F03](#)

## 6.43 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](#)

### 6.43.1 Member Function/Subroutine Documentation

#### 6.43.1.1 print()

Procedure, public mqc\_est::mqc\_wavefunction::print ( )

## 6.43.2 Member Data Documentation

### 6.43.2.1 basis

```
character(len=256) mqc_est::mqc_wavefunction::basis
```

### 6.43.2.2 charge

```
type(mqc_scalar) mqc_est::mqc_wavefunction::charge
```

### 6.43.2.3 core\_hamiltonian

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::core_hamiltonian
```

### 6.43.2.4 density\_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::density_matrix
```

### 6.43.2.5 fock\_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::fock_matrix
```

### 6.43.2.6 mo\_coefficients

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::mo_coefficients
```

#### 6.43.2.7 mo\_energies

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_energies
```

#### 6.43.2.8 mo\_symmetries

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_symmetries
```

#### 6.43.2.9 multiplicity

```
type(mqc_scalar) mqc_est::mqc_wavefunction::multiplicity
```

#### 6.43.2.10 nalpha

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nalpha
```

#### 6.43.2.11 nbasis

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbasis
```

#### 6.43.2.12 nbeta

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbeta
```

#### 6.43.2.13 nelectrons

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nelectrons
```



#### 6.43.2.14 overlap\_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::overlap_matrix
```

#### 6.43.2.15 scf\_density\_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::scf_density_matrix
```

#### 6.43.2.16 symmetry

```
character(len=256) mqc_est::mqc_wavefunction::symmetry
```

#### 6.43.2.17 wf\_complex

```
logical mqc_est::mqc_wavefunction::wf_complex
```

#### 6.43.2.18 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](#)

## 6.44 mqc\_algebra::operator(\*) Interface Reference

**Multiplies two variables**

## 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](#) (Realln, 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, Realln)  
***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)  
***MQC\_ScalarVectorProduct** is a function that returns the product of a MQC scalar with a MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vectorscalarpduct](#) (vector, scalar)  
***MQC\_VectorScalarProduct** is a function that returns the product of a MQC vector with a MQC scalar*
- type([mqc\\_matrix](#)) function [mqc\\_scalarmatrixproduct](#) (Scalar, Matrix)  
 • type([mqc\\_matrix](#)) function [mqc\\_matrixscalarpduct](#) (Matrix, Scalar)  
 • type([mqc\\_vector](#)) function [mqc\\_realvectorproduct](#) (Realln, Vector)  
***MQC\_RealVectorProduct** is a function that returns the product of an intrinsic real scalar and a MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vectorrealproduct](#) (vector, realln)  
***MQC\_VectorRealProduct** is a function that returns the product of a MQC vector and an intrinsic real scalar*
- type([mqc\\_vector](#)) function [mqc\\_integervectorproduct](#) (intIn, Vector)  
***MQC\_IntegerVectorProduct** is a function that returns the product of an intrinsic integer scalar and a MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vectorintegerproduct](#) (vector, intIn)  
***MQC\_VectorIntegerProduct** is a function that returns the product of a MQC vector and an intrinsic integer scalar*
- type([mqc\\_vector](#)) function [mqc\\_complexvectorproduct](#) (ComplexIn, Vector)  
***MQC\_ComplexVectorProduct** is a function that returns the product of an intrinsic complex scalar and a MQC vector*
- type([mqc\\_vector](#)) function [mqc\\_vectorcomplexproduct](#) (vector, complexIn)  
***MQC\_VectorComplexProduct** is a function that returns the product of a MQC vector and an intrinsic complex scalar*
- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixproduct](#) (MA, MB)  
***MQC\_MatrixMatrixProduct** is a function that computes the element- wise product of two MQC matrices*

### 6.44.1 Detailed Description

Multiplies two variables

## 6.44.2 Member Function/Subroutine Documentation

### 6.44.2.1 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

### 6.44.2.2 mqc\_complexvectorproduct()

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

**MQC\_ComplexVectorProduct** is a function that returns the product of an intrinsic complex scalar and a MQC vector

#### Purpose:

MQC\_ComplexVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector.

**Parameters**

in	<i>Comp↔ In</i>	CompIn is Complex(kind=real64) The intrinsic complex to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

L. M. Thompson

**Date**

2019

**6.44.2.3 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>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

#### 6.44.2.4 mqc\_integervectorproduct()

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

**MQC\_IntegerVectorProduct** is a function that returns the product of an intrinsic integer scalar and a MQC vector

Purpose:

MQC\_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector.

Parameters

in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

Author

L. M. Thompson

Date

2019

#### 6.44.2.5 mqc\_matrixmatrixproduct()

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

**MQC\_MatrixMatrixProduct** is a function that computes the element- wise product of two MQC matrices

Purpose:

MQC\_MatrixMatrixProduct is a function that computes the element-wise product of two MQC matrices.

**Parameters**

in	<i>MA</i>	MA is Type(MQC_Matrix) The first MQC matrix.
in	<i>MB</i>	MB is Type(MQC_Matrix) The second MQC matrix.

**Author**

H. P. Hratchian

**Date**

2017

**6.44.2.6 mqc\_matrixscalarproduct()**

```

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

```

**6.44.2.7 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

## 6.44.2.8 mqc\_realvectorproduct()

```

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

```

**MQC\_RealVectorProduct** is a function that returns the product of an intrinsic real scalar and a MQC vector

## Purpose:

MQC\_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector.

## Parameters

in	<i>RealIn</i>	RealIn is Real(kind=real64) The real intrinsic to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

**Author**

L. M. Thompson

**Date**

2019

**6.44.2.9 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>Complex↔ 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



**6.44.2.10 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>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

**6.44.2.11 mqc\_scalarmatrixproduct()**

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

**6.44.2.12 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

**6.44.2.13 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

#### 6.44.2.14 mqc\_scalarvectorproduct()

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

**MQC\_ScalarVectorProduct** is a function that returns the product of a MQC scalar with a MQC vector

Purpose:

MQC\_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector.

Parameters

in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.
in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

Author

X. Sheng

A. D. Mahler

Date

2017, 2019

#### 6.44.2.15 mqc\_vectorcomplexproduct()

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

**MQC\_VectorComplexProduct** is a function that returns the product of a MQC vector and an intrinsic complex scalar

**Purpose:**

`MQC_VectorComplexProduct` is a function that returns the product of a MQC vector and an intrinsic complex scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>CompIn</i>	CompIn is Complex(kind=real64) The integer complex to multiply.

**Author**

L. M. Thompson

**Date**

2019

**6.44.2.16 mqc\_vectorintegerproduct()**

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

**MQC\_VectorIntegerProduct** is a function that returns the product of a MQC vector and an intrinsic integer scalar

**Purpose:**

`MQC_VectorIntegerProduct` is a function that returns the product of a MQC vector and an intrinsic integer scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The integer intrinsic to multiply.

## Author

L. M. Thompson

## Date

2019

## 6.44.2.17 mqc\_vectorrealproduct()

```

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

```

**MQC\_VectorRealProduct** is a function that returns the product of a MQC vector and an intrinsic real scalar

## Purpose:

MQC\_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>RealIn</i>	RealIn is Real(kind=real64) The real intrinsic to multiply.

## Author

L. M. Thompson

## Date

2019

#### 6.44.2.18 mqc\_vectorscalarproduct()

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

**MQC\_VectorScalarProduct** is a function that returns the product of a MQC vector with a MQC scalar

##### Purpose:

MQC\_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar.

##### Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.

##### Author

X. Sheng

##### Date

2017

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

- src/[mqc\\_algebra.F03](#)

## 6.45 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)

#### 6.45.1 Member Function/Subroutine Documentation

6.45.1.1 `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 )
```

6.45.1.2 `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](#)

6.46 `mqc_algebra::operator(**)` Interface Reference

Exponentials a variable to the power of another

## 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\\_scalarintegrexponent](#) (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*

## 6.46.1 Detailed Description

Exponentials a variable to the power of another

## 6.46.2 Member Function/Subroutine Documentation

### 6.46.2.1 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>CompIn</i>	CompIn is Complex(kind=real64) The power value.

#### Author

L. M. Thompson

#### Date

2019

### 6.46.2.2 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

## 6.46.2.3 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

#### 6.46.2.4 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](#)

## 6.47 mqc\_algebra::operator(+) Interface Reference

**Sums two variables**

## 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)  
***MQC\_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector***
- type([mqc\\_vector](#)) function [mqc\\_scalarvectorsum](#) (ScalarIn, VectorIn)  
***MQC\_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector***
- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixsum](#) (MA, MB)  
***MQC\_MatrixMatrixSum is a function that sums two MQC matrices***

### 6.47.1 Detailed Description

Sums two variables

### 6.47.2 Member Function/Subroutine Documentation

#### 6.47.2.1 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>Complex</i> ↔ <i>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

## 6.47.2.2 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</i> ↔ <i>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

### 6.47.2.3 mqc\_matrixmatrixsum()

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

**MQC\_MatrixMatrixSum** is a function that sums two MQC matrices

Purpose:

MQC\_MatrixMatrixSum is a function that sums two MQC matrices.

Parameters

in	<i>MA</i>	MA is Type(MQC_Matrix) First MQC matrix to sum.
in	<i>MB</i>	MB is Type(MQC_Matrix) Second MQC matrix to sum.

Author

H. P. Hratchian

L. M. Thompson

Date

2017, 2018

### 6.47.2.4 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

**6.47.2.5 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

### 6.47.2.6 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

### 6.47.2.7 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

**6.47.2.8 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

**6.47.2.9 mqc\_scalarvectorsum()**

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

**MQC\_ScalarVectorSum** is a function that adds an MQC scalar to all elements of an MQC vector

Purpose:

MQC\_VectorVectorSum is a function that adds an MQC scalar to all elements of an MQC vector.

Parameters

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC scalar to add to the MQC vector.
in	<i>Vector</i> ↔ <i>In</i>	VectorIn is Type(MQC_Vector) The MQC vector with elements to sum with ScalarIn.

Author

L. M. Thompson

Date

2016

**6.47.2.10 mqc\_vectorvectorsum()**

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

**MQC\_VectorVectorSum** is a function that adds two MQC vectors and stores them in another MQC vector

Purpose:

MQC\_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector.

## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector that will be summed.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector that will be summed.

## Author

L. M. Thompson

## Date

2016

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\\_sum](#) (integralA, integralB)

### 6.48.1 Member Function/Subroutine Documentation

#### 6.48.1.1 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](#)

## 6.49 mqc\_est::operator(-) Interface Reference

### Public Member Functions

- type([mqc\\_scf\\_integral](#)) function [mqc\\_integral\\_difference](#) (integralA, integralB)

### 6.49.1 Member Function/Subroutine Documentation

#### 6.49.1.1 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](#)

## 6.50 mqc\_algebra::operator(-) Interface Reference

Subtracts two variables

### 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)  
***MQC\_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector***
- type([mqc\\_vector](#)) function [mqc\\_scalarvectordifference](#) (ScalarIn, VectorIn)  
***MQC\_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector***
- type([mqc\\_matrix](#)) function [mqc\\_matrixmatrixsubtract](#) (MA, MB)  
***MQC\_MatrixMatrixSubtract is a function that subtracts two MQC matrices***

## 6.50.1 Detailed Description

Subtracts two variables

## 6.50.2 Member Function/Subroutine Documentation

### 6.50.2.1 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>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

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

**Author**

L. M. Thompson

**Date**

2019

**6.50.2.3 mqc\_matrixmatrixsubtract()**

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

**MQC\_MatrixMatrixSubtract is a function that subtracts two MQC matrices**

**Purpose:**

`MQC_MatrixMatrixSubtract` is a function that subtracts two MQC matrices.

**Parameters**

in	<i>MA</i>	MA is <code>Type(MQC_Matrix)</code> The matrix that MB will be subtracted from.
in	<i>MB</i>	MB is <code>Type(MQC_Matrix)</code> The matrix that will be subtracted from MA.

**Author**

H. P. Hratchian

**Date**

2017

**6.50.2.4 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

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

**Author**

L. M. Thompson

**Date**

2019

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

**Author**

L. M. Thompson

**Date**

2019

**6.50.2.7 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>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real to subtract.

**Author**

L. M. Thompson

**Date**

2019



**6.50.2.8 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

**6.50.2.9 mqc\_scalarvectordifference()**

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

**MQC\_ScalarVectorDifference** is a function that subtracts an MQC scalar from all elements of an MQC vector

**Purpose:**

MQC\_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector.

## Parameters

in	<i>Scalar</i> ↔ <i>In</i>	ScalarIn is Type(MQC_Scalar) The MQC scalar to be subtracted from elements of the the MQC vector.
in	<i>Vector</i> ↔ <i>In</i>	VectorIn is Type(MQC_Vector) The MQC vector with elements from which ScalarIn will be subtracted.

## Author

L. M. Thompson

## Date

2016

## 6.50.2.10 mqc\_vectorvectordifference()

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

**MQC\_VectorVectorDifference** is a function that subtracts two MQC vectors and stores them in another MQC vector

## Purpose:

MQC\_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector.

## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector from which the second will be subtracted.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector that will be subtracted from the first.

## Author

L. M. Thompson

## Date

2016

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

- [src/mqc\\_algebra.F03](#)

## 6.51 `mqc_algebra::operator(.dot.)` Interface Reference

Computes the inner product of two arrays

### Public Member Functions

- `type(mqc_scalar)` function `mqc_vectorvectordotproduct` (Vector1, Vector2)  
*MQC\_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors*
- `type(mqc_vector)` function `mqc_vectormatrixdotproduct` (VA, MB)
- `type(mqc_vector)` function `mqc_matrixvectordotproduct` (MA, VB)
- `type(mqc_matrix)` function `mqc_matrixmatrixdotproduct` (MA, MB)

### 6.51.1 Detailed Description

Computes the inner product of two arrays

### 6.51.2 Member Function/Subroutine Documentation

#### 6.51.2.1 `mqc_matrixmatrixdotproduct()`

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

### 6.51.2.2 mqc\_matrixvectordotproduct()

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

### 6.51.2.3 mqc\_vectormatrixdotproduct()

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

### 6.51.2.4 mqc\_vectorvectordotproduct()

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

**MQC\_VectorVectorDotProduct** is a function that returns the dot product of two MQC vectors

#### Purpose:

MQC\_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors. The first vector should be a row vector, while the second vector should be a column vector. The vectors should be of the same length.

#### Parameters

in	<i>Vector1</i>	Vector1 is Type(MQC_Vector) The MQC row vector.
in	<i>Vector2</i>	Vector2 is Type(MQC_Vector) The MQC column vector.

#### Author

H. P. Hratchian

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(.eq.) Interface Reference

Determines if two variables are equal

### Public Member Functions

- logical function [mqc\\_scalareq](#) (Scalar1, Scalar2)  
*MQC\_ScalarEQ is a function that returns TRUE if two MQC\_Scalar variables are equal*

### 6.52.1 Detailed Description

Determines if two variables are equal

### 6.52.2 Member Function/Subroutine Documentation

#### 6.52.2.1 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](#)

## 6.53 mqc\_algebra::operator(.ewd.) Interface Reference

Computes the element-wise quotient of two arrays

### Public Member Functions

- type([mqc\\_matrix](#)) function [mqc\\_elementmatrixdivide](#) (A, B)  
*MQC\_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices*

#### 6.53.1 Detailed Description

Computes the element-wise quotient of two arrays

#### 6.53.2 Member Function/Subroutine Documentation

##### 6.53.2.1 mqc\_elementmatrixdivide()

```
type(mqc\_matrix) function mqc_algebra::operator(.ewd.)::mqc_elementmatrixdivide (
    type(mqc\_matrix), intent(in) A,
    type(mqc\_matrix), intent(in) B )
```

**MQC\_ElementMatrixDivide** is a function that returns the element- wise quotient of two MQC matrices

#### Purpose:

MQC\_ElementMatrixDivide is a function that returns the element-wise quotient of two MQC matrices.

## Parameters

in	<i>A</i>	A is type( <code>mqc_matrix</code> ) The matrix with elements being the numerator.
in	<i>B</i>	B is type( <code>mqc_matrix</code> ) The matrix with elements being the denominator.

## Author

X. Sheng

## Date

2017

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

- [src/mqc\\_algebra.F03](#)

## 6.54 `mqc_algebra::operator(.ewp.)` Interface Reference

Computes the element-wise product of two arrays

### Public Member Functions

- type(`mqc_vector`) function `mqc_elementvectorproduct` (Vector1In, Vector2In)  
*MQC\_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector*
- type(`mqc_matrix`) function `mqc_elementmatrixproduct` (A, B)  
*MQC\_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices*

### 6.54.1 Detailed Description

Computes the element-wise product of two arrays

### 6.54.2 Member Function/Subroutine Documentation

### 6.54.2.1 mqc\_elementmatrixproduct()

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

**MQC\_ElementMatrixProduct** is a function that returns the element- wise product of two MQC matrices

#### Purpose:

MQC\_ElementMatrixProduct is a function that returns the element-wise product of two MQC matrices.

#### Parameters

in	<i>A</i>	A is type(mqc_matrix) The first matrix to element-wise multiply.
in	<i>B</i>	B is type(mqc_matrix) The second matrix to element-wise multiply.

#### Author

X. Sheng

#### Date

2017

### 6.54.2.2 mqc\_elementvectorproduct()

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

**MQC\_ElementVectorProduct** is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector

#### Purpose:

MQC\_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector.



## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The frist MQC vector to multiply elementwise.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector to multiply elementwise.

## 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(.ge.) Interface Reference

Determines if a variable is greater than or equal to another

### 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*

#### 6.55.1 Detailed Description

Determines if a variable is greater than or equal to another

#### 6.55.2 Member Function/Subroutine Documentation

### 6.55.2.1 `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](#)

## 6.56 `mqc_algebra::operator(.gt.)` Interface Reference

**Determines if a variable is greater than another**

## 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\\_integertgtscalar](#) (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*

### 6.56.1 Detailed Description

Determines if a variable is greater than another

### 6.56.2 Member Function/Subroutine Documentation

#### 6.56.2.1 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.
Generated by Doxygen		

**Author**

L. M. Thompson

**Date**

2019

**6.56.2.2 mqc\_realgtscalar()**

```
logical function mqc_algebra::operator(.gt.)::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>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

### 6.56.2.3 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

### 6.56.2.4 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( <code>MQC_Scalar</code> ) The <code>MQC_Scalar</code> 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

**6.56.2.5 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>RealIn</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(.le.) Interface Reference

Determines if a variable is less than or equal to another

### 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, 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\\_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\\_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\\_integerlescalar](#) (IntIn, Scalar)  
*MQC\_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC\_Scalar*

### 6.57.1 Detailed Description

Determines if a variable is less than or equal to another

## 6.57.2 Member Function/Subroutine Documentation

### 6.57.2.1 `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

### 6.57.2.2 `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

**6.57.2.3 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

**6.57.2.4 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

6.57.2.5 `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>RealIn</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](#)

## 6.58 mqc\_algebra::operator(.lt.) Interface Reference

Determines if a variable is less than another

### 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***

### 6.58.1 Detailed Description

Determines if a variable is less than another

### 6.58.2 Member Function/Subroutine Documentation

#### 6.58.2.1 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

**6.58.2.2 `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

### 6.58.2.3 mqc\_scalarltreal()

```
logical function mqc_algebra::operator(.lt.)::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

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

- [src/mqc\\_algebra.F03](#)

## 6.59 mqc\_algebra::operator(.ne.) Interface Reference

**Determines if two variables are not equal**

## 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*

### 6.59.1 Detailed Description

Determines if two variables are not equal

### 6.59.2 Member Function/Subroutine Documentation

#### 6.59.2.1 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](#)

## 6.60 mqc\_algebra::operator(.outer.) Interface Reference

Computes the outer product of two vectors

### Public Member Functions

- `type(mqc_matrix)` function `mqc_outer` (VA, VB)  
*MQC\_Outer is a function that returns the outer product of two MQC vectors*

#### 6.60.1 Detailed Description

Computes the outer product of two vectors

#### 6.60.2 Member Function/Subroutine Documentation

##### 6.60.2.1 mqc\_outer()

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

**MQC\_Outer is a function that returns the outer product of two MQC vectors**

**Purpose:**

MQC\_Outer is a function that returns the outer product of two MQC vectors. The first vector should be a column vector, while the second vector should be a row vector.

#### Parameters

in	VA	VA is Type(MQC_Vector) The MQC column vector.
in	VB	VB is Type(MQC_Vector) The MQC row vector.



Author

X. Sheng

Date

2017

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

- [src/mqc\\_algebra.F03](#)

## 6.61 `mqc_algebra::operator(.x.)` Interface Reference

Computes the cross product of two vectors

### Public Member Functions

- `type(mqc_vector)` function `mqc_crossproduct` (Vector1In, Vector2In)  
*MQC\_CrossProduct is a function that returns the cross product of two MQC vectors*

#### 6.61.1 Detailed Description

Computes the cross product of two vectors

#### 6.61.2 Member Function/Subroutine Documentation

##### 6.61.2.1 `mqc_crossproduct()`

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

**MQC\_CrossProduct** is a function that returns the cross product of two MQC vectors

Purpose:

`MQC_CrossProduct` is a function that returns the cross product of two MQC vectors. The vectors should both be of length 3.

## Parameters

in	<i>Vector1</i> ↔ <i>In</i>	Vector1In is Type(MQC_Vector) The first MQC vector.
in	<i>Vector2</i> ↔ <i>In</i>	Vector2In is Type(MQC_Vector) The second MQC vector.

## Author

L. M. Thompson

## Date

2016

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

- src/[mqc\\_algebra.F03](#)

## 6.62 mqc\_algebra::operator(/) Interface Reference

Divides two variables

### 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)  
*MQC\_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar*
- type([mqc\\_vector](#)) function [mqc\\_vectorrealddivide](#) (vector, realIn)  
*MQC\_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer*
- type([mqc\\_vector](#)) function [mqc\\_vectorintegerdivide](#) (vector, intIn)  
*MQC\_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar*
- type([mqc\\_vector](#)) function [mqc\\_vectorcomplexdivide](#) (vector, compln)  
*MQC\_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar*

## 6.62.1 Detailed Description

Divides two variables

## 6.62.2 Member Function/Subroutine Documentation

### 6.62.2.1 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>	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator.

#### Author

L. M. Thompson

#### Date

2019

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

**Author**

L. M. Thompson

**Date**

2019

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

**Author**

L. M. Thompson

**Date**

2019

**6.62.2.4 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

### 6.62.2.5 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

### 6.62.2.6 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 Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	<i>Integer↔ In</i>	IntegerIn is Integer(kind=int64) The intrinsic integer variable denominator.

## Author

L. M. Thompson

## Date

2019

## 6.62.2.7 mqc\_scalarrealdive()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarrealdive (
    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 Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	<i>Real↔ In</i>	RealIn is Real(kind=real64) The intrinsic real variable denominator.

## Author

L. M. Thompson

Date

2019

### 6.62.2.8 mqc\_vectorcomplexdivide()

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

**MQC\_VectorComplexDivide** is a function that returns a MQC vector divided by an intrinsic complex scalar

Purpose:

MQC\_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar.

Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>Comp↔ In</i>	CompIn is Complex(kind=comp64) The intrinsic complex scalar to divide by.

Author

L. M. Thompson

Date

2019

### 6.62.2.9 mqc\_vectorintegerdivide()

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

**MQC\_VectorIntegerDivide** is a function that returns a MQC vector divided by an intrinsic integer scalar

Purpose:

MQC\_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar.



## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>IntIn</i>	IntIn is Integer(kind=int64) The intrinsic integer scalar to divide by.

## Author

L. M. Thompson

## Date

2019

## 6.62.2.10 mqc\_vectorrealdivide()

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

**MQC\_VectorRealDivide** is a function that returns a MQC vector divided by an intrinsic real integer

## Purpose:

MQC\_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real scalar.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>RealIn</i>	RealIn is Real(kind=real64) The intrinsic real scalar to divide by.

## Author

L. M. Thompson

Date

2019

### 6.62.2.11 mqc\_vectorscalardivide()

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

**MQC\_VectorScalarDivide** is a function that returns a MQC vector divided by a MQC scalar

**Purpose:**

MQC\_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar.

**Parameters**

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	<i>Scalar</i>	Scalar is Type(MQC_Scalar) The MQC_Scalar to divide by.

**Author**

A. D. Mahler

Date

2019

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

- src/[mqc\\_algebra.F03](#)

## 6.63 mqc\_algebra::real Interface Reference

**Returns the real part**

## 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)  
***MQC\_Vector\_Complex\_RealPart** is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector*

### 6.63.1 Detailed Description

Returns the real part

### 6.63.2 Member Function/Subroutine Documentation

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

Author

L. M. Thompson

Date

2019

### 6.63.2.2 `mqc_vector_complex_realpart()`

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

**MQC\_Vector\_Complex\_RealPart** is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

#### Purpose:

MQC\_Vector\_Complex\_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector.

#### Parameters

in	<i>A</i>	
		A is Class(MQC_Vector) The name of the MQC_Vector variable.

#### 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::sin` Interface Reference

Returns the sine

### 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*

### 6.64.1 Detailed Description

Returns the sine

## 6.64.2 Member Function/Subroutine Documentation

### 6.64.2.1 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](#)

## 6.65 mqc\_algebra::sqrt Interface Reference

Returns the square root

### 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*

### 6.65.1 Detailed Description

Returns the square root

### 6.65.2 Member Function/Subroutine Documentation

#### 6.65.2.1 `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	<i>Scalar</i>	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](#)

## 6.66 `mqc_algebra::tan` Interface Reference

Returns the tangent

## 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*

### 6.66.1 Detailed Description

Returns the tangent

### 6.66.2 Member Function/Subroutine Documentation

#### 6.66.2.1 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

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`

## 6.67 `mqc_est::transpose` Interface Reference

### Public Member Functions

- type([mqc\\_scf\\_integral](#)) function [mqc\\_integral\\_transpose](#) (integral, label)

### 6.67.1 Member Function/Subroutine Documentation

#### 6.67.1.1 `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](#)

## 6.68 `mqc_algebra::transpose` Interface Reference

Returns the transpose

### Public Member Functions

- type([mqc\\_vector](#)) function [mqc\\_vector\\_transpose](#) (Vector)  
*MQC\_Vector\_Transpose is a function that returns the transpose of an MQC vector*
- type([mqc\\_matrix](#)) function [mqc\\_matrix\\_transpose](#) (Matrix)  
*MQC\_Matrix\_Transpose is a function that returns the transpose of a MQC matrix*

### 6.68.1 Detailed Description

Returns the transpose

### 6.68.2 Member Function/Subroutine Documentation

#### 6.68.2.1 `mqc_matrix_transpose()`

```
type(mqc\_matrix) function mqc_algebra::transpose::mqc_matrix_transpose (
    class(mqc\_matrix), intent(in) Matrix )
```

**MQC\_Matrix\_Transpose** is a function that returns the transpose of a MQC matrix

Purpose:

`MQC_Matrix_Transpose` is a function that returns the transpose of a MQC matrix.



## Parameters

in	<i>Matrix</i>	Matrix is Type(MQC_Matrix) The MQC matrix to be transposed.
----	---------------	--

## Author

L. M. Thompson

X. Sheng

## Date

2016, 2017

## 6.68.2.2 mqc\_vector\_transpose()

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

**MQC\_Vector\_Transpose** is a function that returns the transpose of an MQC vector

## Purpose:

MQC\_Vector\_Transpose is a function that returns the transpose of an MQC vector.

## Parameters

in	<i>Vector</i>	Vector is Type(MQC_Vector) The MQC vector to transpose.
----	---------------	--

## Author

H. P. Hratchian

## Date

2016

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`  
*Rank 0 array variable*
- type `mqc_algebra::mqc_vector`  
*Rank 1 array variable*
- type `mqc_algebra::mqc_matrix`  
*Rank 2 array variable*
- type `mqc_algebra::mqc_r4tensor`  
*Updates the specified element of the MQC Matrix to the specified value*
- interface `mqc_algebra::mqc_print`  
*Prints an object*
- interface `mqc_algebra::contraction`  
*Contracts two arrays*
- interface `mqc_algebra::conjg`  
*Returns the complex conjugate*
- interface `mqc_algebra::mqc_have_real`  
*Determines in an array is real type*
- interface `mqc_algebra::mqc_have_int`  
*Determines in an array is integer type*
- interface `mqc_algebra::mqc_have_complex`  
*Determines in an array is complex type*
- interface `mqc_algebra::mqc_cast_integer`  
*Sets an array to integer type*
- interface `mqc_algebra::mqc_cast_real`  
*Sets an array to real type*
- interface `mqc_algebra::mqc_cast_complex`  
*Sets an array to complex type*
- interface `mqc_algebra::matmul`

***Multiplies two arrays***

- interface [mqc\\_algebra::transpose](#)

***Returns the transpose***

- interface [mqc\\_algebra::dagger](#)

***Returns the Hermitian conjugate***

- interface [mqc\\_algebra::cmplx](#)

***Defines a complex number***

- interface [mqc\\_algebra::sqrt](#)

***Returns the square root***

- interface [mqc\\_algebra::abs](#)

***Takes the absolute value***

- interface [mqc\\_algebra::real](#)

***Returns the real part***

- interface [mqc\\_algebra::aimag](#)

***Returns the imaginary part***

- interface [mqc\\_algebra::sin](#)

***Returns the sine***

- interface [mqc\\_algebra::cos](#)

***Returns the cosine***

- interface [mqc\\_algebra::tan](#)

***Returns the tangent***

- interface [mqc\\_algebra::asin](#)

***Returns the arcsine***

- interface [mqc\\_algebra::acos](#)

***Returns the arccosine***

- interface [mqc\\_algebra::atan](#)

***Returns the arctangent***

- interface [mqc\\_algebra::atan2](#)

***Returns the arctangent accounting for circle quadrant***

- interface [mqc\\_algebra::mqc\\_set\\_array2vector](#)

***Sets an intrinsic array as an MQC Algebra object***

- interface [mqc\\_algebra::mqc\\_matrix\\_symmmatrix\\_put](#)

***Sets a symmetric packed intrinsic array as an MQC Matrix object***

- interface [mqc\\_algebra::mqc\\_matrix\\_diagmatrix\\_put](#)

***Sets a diagonal packed intrinsic array as an MQC Matrix object***

- interface [mqc\\_algebra::matrix\\_symm2sq](#)

***Sets a symmetric packed intrinsic array as a square packed intrinsic array***

- interface [mqc\\_algebra::dot\\_product](#)

***Returns the dot product***

- interface [mqc\\_algebra::assignment\(=\)](#)

***Assigns a variable to the value of another***

- interface [mqc\\_algebra::operator\(+\)](#)

***Sums two variables***

- interface [mqc\\_algebra::operator\(-\)](#)

***Subtracts two variables***

- interface [mqc\\_algebra::operator\(\\*\)](#)

***Multiplies two variables***

- interface `mqc_algebra::operator(/)`  
*Divides two variables*
- interface `mqc_algebra::operator(**)`  
*Exponentials a variable to the power of another*
- interface `mqc_algebra::operator(.ne.)`  
*Determines if two variables are not equal*
- interface `mqc_algebra::operator(.eq.)`  
*Determines if two variables are equal*
- interface `mqc_algebra::operator(.lt.)`  
*Determines if a variable is less than another*
- interface `mqc_algebra::operator(.gt.)`  
*Determines if a variable is greater than another*
- interface `mqc_algebra::operator(.le.)`  
*Determines if a variable is less than or equal to another*
- interface `mqc_algebra::operator(.ge.)`  
*Determines if a variable is greater than or equal to another*
- interface `mqc_algebra::assignment(=)`  
*Assigns a variable to the value of another*
- interface `mqc_algebra::operator(.dot.)`  
*Computes the inner product of two arrays*
- interface `mqc_algebra::operator(*)`  
*Multiplies two variables*
- interface `mqc_algebra::operator(/)`  
*Divides two variables*
- interface `mqc_algebra::operator(+)`  
*Sums two variables*
- interface `mqc_algebra::operator(-)`  
*Subtracts two variables*
- interface `mqc_algebra::operator(.ewp.)`  
*Computes the element-wise product of two arrays*
- interface `mqc_algebra::operator(.ewd.)`  
*Computes the element-wise quotient of two arrays*
- interface `mqc_algebra::operator(.x.)`  
*Computes the cross product of two vectors*
- interface `mqc_algebra::operator(.outer.)`  
*Computes the outer product of two vectors*
- interface `mqc_algebra::assignment(=)`  
*Assigns a variable to the value of another*
- interface `mqc_algebra::operator(+)`  
*Sums two variables*
- interface `mqc_algebra::operator(-)`  
*Subtracts two variables*
- interface `mqc_algebra::operator(*)`  
*Multiplies two variables*
- interface `mqc_algebra::operator(.dot.)`  
*Computes the inner product of two arrays*
- interface `mqc_algebra::assignment(=)`  
*Assigns a variable to the value of another*

## Modules

- module [mqc\\_algebra](#)

*MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran*

## 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, Seed, Distribution)  
***MQC\_Scalar\_Get\_Random\_Value** is a function that returns a random real value from a specified distribution*
- 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, Realln)  
***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](#) (Realln, 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\\_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\\_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)  
***MQC\_Allocate\_Vector** is used to allocate a vector type variable of the MQC\_Vector class*
- subroutine [mqc\\_algebra::mqc\\_deallocate\\_vector](#) (Vector)  
***MQC\_Deallocate\_Vector** is used to deallocate a vector type variable of the MQC\_Vector class*
- integer(kind=int64) function [mqc\\_algebra::mqc\\_length\\_vector](#) (Vector)  
***MQC\_Length\_Vector** is used to return the length of an MQC vector*
- logical function [mqc\\_algebra::mqc\\_vector\\_havereal](#) (Vector)  
***MQC\_Vector\_HaveReal** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector*
- logical function [mqc\\_algebra::mqc\\_vector\\_haveinteger](#) (Vector)  
***MQC\_Vector\_HaveInteger** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector*
- logical function [mqc\\_algebra::mqc\\_vector\\_havecomplex](#) (Vector)  
***MQC\_Vector\_HaveComplex** is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector*
- logical function [mqc\\_algebra::mqc\\_vector\\_iscolumn](#) (Vector)

***MQC\_Vector\_IsColumn*** is a function that returns **TRUE** if the **MQC** vector is a column vector and **FALSE** if the **MQC** vector is a row vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_int2real](#) (Vector)

***MQC\_Vector\_Copy\_Int2Real*** is a subroutine that copies an integer **MQC\_Vector** into its real vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_int2complex](#) (Vector)

***MQC\_Vector\_Copy\_Int2Complex*** is a subroutine that copies an integer **MQC\_Vector** into its complex vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_real2int](#) (Vector)

***MQC\_Vector\_Copy\_Real2Int*** is a subroutine that copies a real **MQC\_Vector** into its integer vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_real2complex](#) (Vector)

***MQC\_Vector\_Copy\_Real2Complex*** is a subroutine that copies a real **MQC\_Vector** into its complex vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_complex2int](#) (Vector)

***MQC\_Vector\_Copy\_Complex2Int*** is a subroutine that copies a complex **MQC\_Vector** into its integer vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_copy\\_complex2real](#) (Vector)

***MQC\_Vector\_Copy\_Complex2Real*** is a subroutine that copies a complex **MQC\_Vector** into its real vector

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_scalar\\_at](#) (Vec, I)

***MQC\_Vector\_Scalar\_At*** is a function that returns the *ith* element of a **MQC** vector as an **MQC** scalar

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_vector\\_at](#) (Vec, I, J)

***MQC\_Vector\_Vector\_At*** is a function that returns the vector at the specified subvector of **MQC\_Vector**

- subroutine [mqc\\_algebra::mqc\\_set\\_vector2integerarray](#) (ArrayOut, VectorIn)

***MQC\_Set\_Vector2IntegerArray*** is a subroutine that outputs an **MQC** vector to a rank 1 intrinsic integer array

- subroutine [mqc\\_algebra::mqc\\_set\\_vector2realarray](#) (ArrayOut, VectorIn)

***MQC\_Set\_Vector2RealArray*** is a subroutine that outputs an **MQC** vector to a rank 1 intrinsic real array

- subroutine [mqc\\_algebra::mqc\\_set\\_vector2complexarray](#) (ArrayOut, VectorIn)

***MQC\_Set\_Vector2ComplexArray*** is a subroutine that outputs an **MQC** vector to a rank 1 intrinsic complex array

- subroutine [mqc\\_algebra::mqc\\_set\\_array2vector\\_integer](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Integer*** is a subroutine that sets a rank 1 intrinsic integer array equal to a **MQC** vector

- subroutine [mqc\\_algebra::mqc\\_set\\_array2vector\\_real](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Real*** is a subroutine that sets a rank 1 vector intrinsic real array equal to a **MQC** vector

- subroutine [mqc\\_algebra::mqc\\_set\\_array2vector\\_complex](#) (VectorOut, ArrayIn)

***MQC\_Set\_Array2Vector\_Complex*** is a subroutine that sets a rank 1 vector intrinsic complex array equal to a **MQC** vector

- subroutine [mqc\\_algebra::mqc\\_set\\_vector2vector](#) (VectorOut, VectorIn)

***MQC\_Set\_Vector2Vector*** is a subroutine that sets a **MQC** vector equal to another **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorvectorsum](#) (Vector1In, Vector2In)

***MQC\_VectorVectorSum*** is a function that adds two **MQC** vectors and stores them in another **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorvectordifference](#) (Vector1In, Vector2In)

***MQC\_VectorVectorDifference*** is a function that subtracts two **MQC** vectors and stores them in another **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_scalarvectorsum](#) (ScalarIn, VectorIn)

***MQC\_ScalarVectorSum*** is a function that adds an **MQC** scalar to all elements of an **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_scalarvectordifference](#) (ScalarIn, VectorIn)

***MQC\_ScalarVectorDifference*** is a function that subtracts an **MQC** scalar from all elements of an **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_elementvectorproduct](#) (Vector1In, Vector2In)

***MQC\_ElementVectorProduct*** is a function that multiplies two **MQC** vectors elementwise and stores them into another **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_transpose](#) (Vector)

***MQC\_Vector\_Transpose*** is a function that returns the transpose of an **MQC** vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_conjugate\\_transpose](#) (Vector)

***MQC\_Vector\_Conjugate\_Transpose*** is a function that returns the conjugate transpose of an **MQC** vector

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vectorvectordotproduct](#) (Vector1, Vector2)  
***MQC\_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors***
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_outer](#) (VA, VB)  
***MQC\_Outer is a function that returns the outer product of two MQC vectors***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_crossproduct](#) (Vector1In, Vector2In)  
***MQC\_CrossProduct is a function that returns the cross product of two MQC vectors***
- subroutine [mqc\\_algebra::mqc\\_print\\_vector\\_algebra1](#) (Vector, IOut, Header, Verbose, Blank\_At\_Top, Blank\_At\_Bottom)  
***MQC\_Print\_Vector\_Algebra1 is a subroutine used to print an MQC vector***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_cast\\_integer](#) (VA)  
***MQC\_vector\_cast\_integer is a function that converts an MQC vector to its integer space***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_cast\\_real](#) (VA)  
***MQC\_vector\_cast\_real is a function that converts an MQC vector to its real space***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_cast\\_complex](#) (VA)  
***MQC\_vector\_cast\_complex is a function that converts an MQC vector to its complex space***
- subroutine [mqc\\_algebra::mqc\\_vector\\_scalar\\_put](#) (Vector, Scalar, I)  
***MQC\_Vector\_Scalar\_Put is a subroutine that updates the value of the ith element of a MQC vector with the value of a MQC scalar***
- subroutine [mqc\\_algebra::mqc\\_vector\\_scalar\\_increment](#) (Vector, Scalar, I)  
***MQC\_Vector\_Scalar\_Increment is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar***
- subroutine [mqc\\_algebra::mqc\\_vector\\_vector\\_put](#) (Vector, VectorIn, I)  
***MQC\_Vector\_Vector\_Put is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector***
- subroutine [mqc\\_algebra::mqc\\_vector\\_initialize](#) (Vector, Length, Scalar)  
***MQC\_Vector\_Initialize is a subroutine that initializes a MQC vector***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_scalarvectorproduct](#) (Scalar, Vector)  
***MQC\_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorscalarproduct](#) (vector, scalar)  
***MQC\_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorscalardivide](#) (vector, scalar)  
***MQC\_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_realvectorproduct](#) (RealIn, Vector)  
***MQC\_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorrealproduct](#) (vector, realIn)  
***MQC\_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorrealdive](#) (vector, realIn)  
***MQC\_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_integervectorproduct](#) (intIn, Vector)  
***MQC\_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorintegerproduct](#) (vector, intIn)  
***MQC\_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorintegerdivide](#) (vector, intIn)  
***MQC\_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar***
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_complexvectorproduct](#) (Compln, Vector)

***MQC\_ComplexVectorProduct*** is a function that returns the product of an intrinsic complex scalar and a MQC vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorcomplexproduct](#) (vector, compln)

***MQC\_VectorComplexProduct*** is a function that returns the product of a MQC vector and an intrinsic complex scalar

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vectorcomplexdivide](#) (vector, compln)

***MQC\_VectorComplexDivide*** is a function that returns a MQC vector divided by an intrinsic complex scalar

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_norm](#) (vector, methodIn)

***MQC\_Vector\_Norm*** is a function that returns the norm of an MQC vector

- logical function [mqc\\_algebra::mqc\\_vector\\_isallocated](#) (Vector)

***MQC\_Vector\_isAllocated*** is a function that returns **TRUE** if an MQC vector is allocated and **FALSE** if it is not

- subroutine [mqc\\_algebra::mqc\\_vector\\_push](#) (Vector, Scalar)

***MQC\_Vector\_Push*** is a function that adds a value to the end of a MQC vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_unshift](#) (Vector, Scalar)

***MQC\_Vector\_Unshift*** is a function that adds a value to the beginning of a MQC vector

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_pop](#) (Vector)

***MQC\_Vector\_Pop*** is a function that removes a value from the end of a MQC vector and returns it

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_shift](#) (Vector)

***MQC\_Vector\_Shift*** is a function that removes a value from the beginning of a MQC vector and returns it

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_maxval](#) (Vector)

***MQC\_Vector\_MaxVal*** is a function that returns the largest value in an MQC vector

- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_vector\\_minval](#) (Vector)

***MQC\_Vector\_MinVal*** is a function that returns the smallest value in an MQC vector

- integer function [mqc\\_algebra::mqc\\_vector\\_maxloc](#) (Vector)

***MQC\_Vector\_MaxLoc*** is a function that returns the index of the largest value in an MQC vector

- integer function [mqc\\_algebra::mqc\\_vector\\_minloc](#) (Vector)

***MQC\_Vector\_MinLoc*** is a function that returns the index of the smallest value in an MQC vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_argsort](#) (Vector)

***MQC\_Vector\_Argsort*** is a function that returns the indices of an MQC vector sorted from low to high

- subroutine [mqc\\_algebra::mqc\\_vector\\_sort](#) (Vector, idx)

***MQC\_Vector\_Sort*** is a function that returns an MQC vector sorted from low to high unless optional index order is present

- subroutine [mqc\\_algebra::mqc\\_vector\\_sqrt](#) (A)

***MQC\_Vector\_Sqrt*** is a function that returns the square root of all elements of an MQC vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_abs](#) (A)

***MQC\_Vector\_Abs*** is a function that returns the absolute value of all elements of an MQC vector

- subroutine [mqc\\_algebra::mqc\\_vector\\_power](#) (A, P)

***MQC\_Vector\_Power*** is a function that returns the value of all elements of an MQC vector raised to a power

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_complex\\_realpart](#) (A)

***MQC\_Vector\_Complex\_RealPart*** is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_complex\\_imagpart](#) (A)

***MQC\_Vector\_Complex\_ImagPart*** is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

- type(mqc\_vector) function [mqc\\_algebra::mqc\\_vector\\_cmplx](#) (Vector1, Vector2)

***MQC\_Vector\_Cmplx*** is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

- subroutine [mqc\\_algebra::mqc\\_matrix\\_diagonalize](#) (A, EVals, EVecs)  
***MQC\_Matrix\_Diagonalize** is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_cast\\_real](#) (MA)  
***MQC\_Matrix\_Cast\_Real** is a function that converts an MQC matrix to its real space*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_cast\\_integer](#) (MA)  
***MQC\_Matrix\_Cast\_Integer** is a function that converts an MQC matrix to its integer space*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_cast\\_complex](#) (MA)  
***MQC\_Matrix\_Cast\_Complex** is a function that converts an MQC matrix to its complex space*
- type(mqc\_scalar) function [mqc\\_algebra::mqc\\_matrix\\_scalar\\_at](#) (Mat, I, J)  
***MQC\_Matrix\_Scalar\_At** is a function that returns the value of an element of a MQC matrix*
- type(mqc\_vector) function [mqc\\_algebra::mqc\\_matrix\\_vector\\_at](#) (Mat, Rows, Cols)  
***MQC\_Matrix\_Vector\_At** is a function that returns the subvector of an MQC matrix*
- recursive subroutine [mqc\\_algebra::mqc\\_matrix\\_vector\\_put](#) (Mat, VectorIn, Rows, Cols)  
***MQC\_Matrix\_Vector\_Put** is a subroutine that writes a subvector to the specified position of a MQC matrix*
- 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)  
***MQC\_Matrix\_DiagMatrix\_Put\_Vector** is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_diagmatrix\\_put\\_integer](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_Integer** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_diagmatrix\\_put\\_real](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_Real** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_diagmatrix\\_put\\_complex](#) (mat, diagMatrixIn)  
***MQC\_Matrix\_DiagMatrix\_Put\_Complex** is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_symmmatrix\\_put\\_integer](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Integer** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_symmmatrix\\_put\\_real](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Real** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_symmmatrix\\_put\\_complex](#) (mat, symmMatrixIn)  
***MQC\_Matrix\_SymmMatrix\_Put\_Complex** is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector*
- recursive subroutine [mqc\\_algebra::mqc\\_matrix\\_matrix\\_put](#) (Mat, MatrixIn, Rows, Cols)  
***MQC\_Matrix\_Matrix\_Put** is a subroutine that writes a submatrix to the specified position of a MQC matrix*
- integer(kind=int64) function [mqc\\_algebra::symindexhash](#) (i, j, k, l)  
***SymIndexHash** is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_elementmatrixproduct](#) (A, B)  
***MQC\_ElementMatrixProduct** is a function that returns the element- wise product of two MQC matrices*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_elementmatrixdivide](#) (A, B)  
***MQC\_ElementMatrixDivide** is a function that returns the element- wise quotient of two MQC matrices*
- logical function [mqc\\_algebra::mqc\\_matrix\\_test\\_symmetric](#) (Matrix, Option)



***MQC\_Matrix\_Test\_Symmetric* is a function that tests a MQC matrix for symmetry**

- logical function [mqc\\_algebra::mqc\\_matrix\\_test\\_diagonal](#) (Matrix)

***MQC\_Matrix\_Test\_Diagonal* is a function that tests a MQC matrix to determine if it is diagonal**

- subroutine [mqc\\_algebra::mqc\\_allocate\\_matrix](#) (M, N, Matrix, Data\_Type, Storage)

***MQC\_Allocate\_Matrix* is used to allocate a matrix type variable of the MQC\_Matrix class**

- subroutine [mqc\\_algebra::mqc\\_deallocate\\_matrix](#) (Matrix)

***MQC\_Deallocate\_Matrix* is used to deallocate a matrix type variable of the MQC\_Matrix class**

- logical function [mqc\\_algebra::mqc\\_matrix\\_isallocated](#) (Matrix)

***MQC\_Matrix\_isAllocate* is a function that returns the allocation status of a MQC\_Matrix variable**

- subroutine [mqc\\_algebra::mqc\\_set\\_integerarray2matrix](#) (MatrixOut, ArrayIn)

***MQC\_Set\_IntegerArray2Matrix* is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array**

- subroutine [mqc\\_algebra::mqc\\_set\\_realarray2matrix](#) (MatrixOut, ArrayIn)

***MQC\_Set\_RealArray2Matrix* is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array**

- subroutine [mqc\\_algebra::mqc\\_set\\_complexarray2matrix](#) (MatrixOut, ArrayIn)

***MQC\_Set\_ComplexArray2Matrix* is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array**

- subroutine [mqc\\_algebra::mqc\\_set\\_matrix2integerarray](#) (ArrayOut, MatrixIn)

***MQC\_Set\_Matrix2IntegerArray* is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix**

- subroutine [mqc\\_algebra::mqc\\_set\\_matrix2realarray](#) (ArrayOut, MatrixIn)

***MQC\_Set\_Matrix2RealArray* is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix**

- subroutine [mqc\\_algebra::mqc\\_set\\_matrix2complexarray](#) (ArrayOut, MatrixIn)

***MQC\_Set\_Matrix2ComplexArray* is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix**

- subroutine [mqc\\_algebra::mqc\\_set\\_matrix2matrix](#) (MatrixOut, MatrixIn)

***MQC\_Set\_Matrix2Matrix* is a subroutine that sets an MQC matrix equal to another MQC matrix**

- subroutine [mqc\\_algebra::mqc\\_print\\_matrix\\_algebra1](#) (Matrix, IOOut, Header, Blank\_At\_Top, Blank\_At\_Bottom)

***MQC\_Print\_Matrix\_Algebra1* is a subroutine used to print an MQC matrix**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_int2real](#) (Matrix)

***MQC\_Matrix\_Copy\_Int2Real* is a subroutine used to copy an integer MQC matrix into its real space**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_int2complex](#) (Matrix)

***MQC\_Matrix\_Copy\_Int2Complex* is a subroutine used to copy an integer MQC matrix into its complex space**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_real2int](#) (Matrix)

***MQC\_Matrix\_Copy\_Real2Int* is a subroutine used to copy a real MQC matrix into its integer space**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_real2complex](#) (Matrix)

***MQC\_Matrix\_Copy\_Real2Complex* is a subroutine used to copy a real MQC matrix into its complex space**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_complex2int](#) (Matrix)

***MQC\_Matrix\_Copy\_Complex2Int* is a subroutine used to copy a complex MQC matrix into its integer space**

- subroutine [mqc\\_algebra::mqc\\_matrix\\_copy\\_complex2real](#) (Matrix)

***MQC\_Matrix\_Copy\_Complex2Real* is a subroutine used to copy a complex MQC matrix into its real space**

- integer(kind=int64) function [mqc\\_algebra::mqc\\_matrix\\_rows](#) (Matrix)

***MQC\_Matrix\_Rows* is a function used to return the number of rows of an MQC matrix**

- integer(kind=int64) function [mqc\\_algebra::mqc\\_matrix\\_columns](#) (Matrix)

***MQC\_Matrix\_Columns* is a function used to return the number of columns of an MQC matrix**

- logical function [mqc\\_algebra::mqc\\_matrix\\_havereal](#) (Matrix)

***MQC\_Matrix\_HaveReal* is a function used to indicate if an MQC matrix has an allocated real matrix**

- logical function [mqc\\_algebra::mqc\\_matrix\\_haveinteger](#) (Matrix)

***MQC\_Matrix\_HaveInteger* is a function used to indicate if an MQC matrix has an allocated integer matrix**

- logical function [mqc\\_algebra::mqc\\_matrix\\_havecomplex](#) (Matrix)  
***MQC\_Matrix\_HaveComplex** is a function used to indicate if an MQC matrix has an allocated complex matrix*
- logical function [mqc\\_algebra::mqc\\_matrix\\_havefull](#) (Matrix)  
***MQC\_Matrix\_HaveFull** is a function used to indicate if an MQC matrix is stored unpacked*
- logical function [mqc\\_algebra::mqc\\_matrix\\_havesymmetric](#) (Matrix)  
***MQC\_Matrix\_HaveSymmetric** is a function used to indicate if an MQC matrix is stored symmetric-packed*
- logical function [mqc\\_algebra::mqc\\_matrix\\_havediagonal](#) (Matrix)  
***MQC\_Matrix\_HaveDiagonal** is a function used to indicate if an MQC matrix is stored diagonal-packed*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_transpose](#) (Matrix)  
***MQC\_Matrix\_Transpose** is a function that returns the transpose of a MQC matrix*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_conjugate\\_transpose](#) (Matrix)  
***MQC\_Matrix\_Conjugate\_Transpose** is a function that returns the conjugate transpose of a MQC matrix*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_symmetrize](#) (Matrix)  
***MQC\_Matrix\_Symmetrize** is a function that symmetrizes a MQC matrix*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_full2symm](#) (Matrix)  
***MQC\_Matrix\_Full2Symm** is a subroutine that converts an unpacked MQC matrix to symmetric-packed*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_symm2full](#) (Matrix, Option)  
***MQC\_Matrix\_Symm2Full** is a subroutine that converts a symmetry-packed MQC matrix to unpacked*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_full2diag](#) (Matrix)  
***MQC\_Matrix\_Full2Diag** is a subroutine that converts an unpacked MQC matrix to diagonal-packed*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_diag2full](#) (Matrix)  
***MQC\_Matrix\_Diag2Full** is a subroutine that converts a diagonal-packed MQC matrix to unpacked*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_symm2diag](#) (Matrix)  
***MQC\_Matrix\_Symm2Diag** is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed*
- subroutine [mqc\\_algebra::mqc\\_matrix\\_diag2symm](#) (Matrix)  
***MQC\_Matrix\_Diag2Symm** is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrix\\_symm2full\\_func](#) (Matrix)  
***MQC\_Matrix\_Symm2Full\_Func** is a function that converts a symmetric- packed MQC matrix to unpacked*
- subroutine [mqc\\_algebra::matrix\\_symm2sq\\_integer](#) (N, I\_Symm, I\_Sq)  
***Matrix\_Symm2Sq\_Integer** is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array*
- subroutine [mqc\\_algebra::matrix\\_symm2sq\\_real](#) (N, A\_Symm, A\_Sq)  
***Matrix\_Symm2Sq\_Real** is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array*
- subroutine [mqc\\_algebra::matrix\\_symm2sq\\_complex](#) (N, A\_Symm, A\_Sq)  
***Matrix\_Symm2Sq\_Complex** is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_vector2diagmatrix](#) (vector)  
***MQC\_Vector2DiagMatrix** is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrixmatrixsum](#) (MA, MB)  
***MQC\_MatrixMatrixSum** is a function that sums two MQC matrices*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrixmatrixsubtract](#) (MA, MB)  
***MQC\_MatrixMatrixSubtract** is a function that subtracts two MQC matrices*
- type(mqc\_matrix) function [mqc\\_algebra::mqc\\_matrixmatrixproduct](#) (MA, MB)  
***MQC\_MatrixMatrixProduct** is a function that computes the element- wise product of two MQC matrices*
- 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](#) (IOOut, IPrint, NBasisIn, NAlphaIn, NBetaIn, Determinants, NCoreIn)
- type(mqc\_scalar) function [mqc\\_est::slater\\_condon](#) (IOOut, IPrint, NBasisIn, Determinants, L\_A\_String, L\_B\_String, R\_A\_String, R\_B\_String, Core\_Hamiltonian, ERIs, UHF)
- subroutine [mqc\\_est::twoeri\\_trans](#) (IOOut, IPrint, MO\_Coeff, ERIs, MO\_ERIs, UHF)
- subroutine [mqc\\_est::mqc\\_build\\_ci\\_hamiltonian](#) (IOOut, 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, [271](#)
  - mqc\_algebra::mqc\_vector, [281](#)
- addlabel
  - mqc\_est::mqc\_scf\_eigenvalues, [273](#)
  - mqc\_est::mqc\_scf\_integral, [274](#)
- alpha
  - mqc\_est::mqc\_determinant\_string, [245](#)
- argsort
  - mqc\_algebra::mqc\_vector, [281](#)
- at
  - mqc\_algebra::mqc\_matrix, [252](#)
  - mqc\_algebra::mqc\_r4tensor, [270](#)
  - mqc\_algebra::mqc\_vector, [282](#)
  - mqc\_est::mqc\_scf\_eigenvalues, [273](#)
- basis
  - mqc\_est::mqc\_wavefunction, [287](#)
- beta
  - mqc\_est::mqc\_determinant\_string, [245](#)
- bin\_coeff
  - mqc\_algebra, [24](#)
- charge
  - mqc\_est::mqc\_wavefunction, [287](#)
- core\_hamiltonian
  - mqc\_est::mqc\_wavefunction, [287](#)
- cval
  - mqc\_algebra::mqc\_scalar, [271](#)
- dagger
  - mqc\_algebra::mqc\_matrix, [252](#)
  - mqc\_algebra::mqc\_vector, [282](#)
- deleteelist
  - mqc\_est::mqc\_scf\_integral, [274](#)
- density\_matrix
  - mqc\_est::mqc\_wavefunction, [287](#)
- det
  - mqc\_algebra::mqc\_matrix, [252](#)
  - mqc\_est::mqc\_scf\_integral, [274](#)
- diag
  - mqc\_algebra::mqc\_matrix, [253](#)
  - mqc\_algebra::mqc\_vector, [282](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- eigensys
  - mqc\_algebra::mqc\_matrix, [253](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- factorial
  - mqc\_algebra, [25](#)
- fock\_matrix
  - mqc\_est::mqc\_wavefunction, [287](#)
- gen\_det\_str
  - mqc\_est, [183](#)
- get\_one\_gamma\_matrix
  - mqc\_est, [184](#)
- getblock
  - mqc\_est::mqc\_scf\_eigenvalues, [273](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- getelist
  - mqc\_est::mqc\_scf\_integral, [275](#)
- getlabel
  - mqc\_est::mqc\_scf\_eigenvalues, [273](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- identity
  - mqc\_algebra::mqc\_matrix, [253](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- init
  - mqc\_algebra::mqc\_matrix, [253](#)
  - mqc\_algebra::mqc\_r4tensor, [270](#)
  - mqc\_algebra::mqc\_vector, [282](#)
  - mqc\_est::mqc\_scf\_integral, [275](#)
- inv
  - mqc\_algebra::mqc\_matrix, [253](#)
  - mqc\_est::mqc\_scf\_integral, [276](#)
- ival
  - mqc\_algebra::mqc\_scalar, [272](#)
- mat
  - mqc\_algebra::mqc\_matrix, [253](#)
- matrix\_symm2sq\_complex
  - mqc\_algebra, [25](#)
  - mqc\_algebra::matrix\_symm2sq, [236](#)
- matrix\_symm2sq\_integer
  - mqc\_algebra, [26](#)
  - mqc\_algebra::matrix\_symm2sq, [237](#)
- matrix\_symm2sq\_real
  - mqc\_algebra, [27](#)
  - mqc\_algebra::matrix\_symm2sq, [238](#)

- maxloc
  - mqc\_algebra::mqc\_vector, 282
- maxval
  - mqc\_algebra::mqc\_vector, 282
- minloc
  - mqc\_algebra::mqc\_vector, 283
- minval
  - mqc\_algebra::mqc\_vector, 283
- mo\_coefficients
  - mqc\_est::mqc\_wavefunction, 287
- mo\_energies
  - mqc\_est::mqc\_wavefunction, 287
- mo\_symmetries
  - mqc\_est::mqc\_wavefunction, 288
- mput
  - mqc\_algebra::mqc\_matrix, 254
- mqc\_algebra, 11
  - bin\_coeff, 24
  - factorial, 25
  - matrix\_symm2sq\_complex, 25
  - matrix\_symm2sq\_integer, 26
  - matrix\_symm2sq\_real, 27
  - mqc\_allocate\_matrix, 28
  - mqc\_allocate\_r4tensor, 29
  - mqc\_allocate\_scalar, 29
  - mqc\_allocate\_vector, 30
  - mqc\_complexscalaradd, 31
  - mqc\_complexscalardivide, 32
  - mqc\_complexscalarmultiply, 33
  - mqc\_complexscalarsubtract, 33
  - mqc\_complexvectorproduct, 34
  - mqc\_crossproduct, 35
  - mqc\_deallocate\_matrix, 35
  - mqc\_deallocate\_r4tensor, 36
  - mqc\_deallocate\_scalar, 36
  - mqc\_deallocate\_vector, 37
  - mqc\_elementmatrixdivide, 37
  - mqc\_elementmatrixproduct, 38
  - mqc\_elementvectorproduct, 39
  - mqc\_givens\_matrix, 40
  - mqc\_input\_complex\_scalar, 40
  - mqc\_input\_integer\_scalar, 41
  - mqc\_input\_real\_scalar, 41
  - mqc\_integertscalar, 42
  - mqc\_integerlescalar, 43
  - mqc\_integerscalaradd, 43
  - mqc\_integerscalardivide, 44
  - mqc\_integerscalarmultiply, 45
  - mqc\_integerscalarsubtract, 46
  - mqc\_integervectorproduct, 46
  - mqc\_length\_vector, 47
  - mqc\_matrix\_cast\_complex, 47
  - mqc\_matrix\_cast\_integer, 48
  - mqc\_matrix\_cast\_real, 49
  - mqc\_matrix\_columns, 49
  - mqc\_matrix\_conjugate\_transpose, 50
  - mqc\_matrix\_copy\_complex2int, 50
  - mqc\_matrix\_copy\_complex2real, 51
  - mqc\_matrix\_copy\_int2complex, 52
  - mqc\_matrix\_copy\_int2real, 52
  - mqc\_matrix\_copy\_real2complex, 53
  - mqc\_matrix\_copy\_real2int, 53
  - mqc\_matrix\_determinant, 54
  - mqc\_matrix\_diag2full, 54
  - mqc\_matrix\_diag2symm, 55
  - mqc\_matrix\_diagmatrix\_put\_complex, 55
  - mqc\_matrix\_diagmatrix\_put\_integer, 56
  - mqc\_matrix\_diagmatrix\_put\_real, 57
  - mqc\_matrix\_diagmatrix\_put\_vector, 58
  - mqc\_matrix\_diagonalize, 58
  - mqc\_matrix\_full2diag, 59
  - mqc\_matrix\_full2symm, 60
  - mqc\_matrix\_generalized\_eigensystem, 60
  - mqc\_matrix\_havecomplex, 61
  - mqc\_matrix\_havediagonal, 61
  - mqc\_matrix\_havetfull, 62
  - mqc\_matrix\_haveinteger, 62
  - mqc\_matrix\_havereal, 63
  - mqc\_matrix\_havesymmetric, 64
  - mqc\_matrix\_identity, 64
  - mqc\_matrix\_initialize, 64
  - mqc\_matrix\_inverse, 65
  - mqc\_matrix\_isallocated, 65
  - mqc\_matrix\_matrix\_at, 65
  - mqc\_matrix\_matrix\_contraction, 66
  - mqc\_matrix\_matrix\_put, 66
  - mqc\_matrix\_norm, 68
  - mqc\_matrix\_rms\_max, 68
  - mqc\_matrix\_rows, 68
  - mqc\_matrix\_scalar\_at, 69
  - mqc\_matrix\_scalar\_put, 69
  - mqc\_matrix\_set, 70
  - mqc\_matrix\_sqrt, 70
  - mqc\_matrix\_svd, 70
  - mqc\_matrix\_symm2diag, 70
  - mqc\_matrix\_symm2full, 71
  - mqc\_matrix\_symm2full\_func, 72
  - mqc\_matrix\_symmetrize, 72
  - mqc\_matrix\_symmmatrix\_put\_complex, 73
  - mqc\_matrix\_symmmatrix\_put\_integer, 74
  - mqc\_matrix\_symmmatrix\_put\_real, 74
  - mqc\_matrix\_symmsymmr4tensor\_put\_complex, 75
  - mqc\_matrix\_symmsymmr4tensor\_put\_real, 75
  - mqc\_matrix\_test\_diagonal, 75
  - mqc\_matrix\_test\_symmetric, 76
  - mqc\_matrix\_trace, 77
  - mqc\_matrix\_transpose, 77
  - mqc\_matrix\_vector\_at, 77

mqc\_matrix\_vector\_put, 78  
mqc\_matrixmatrixdotproduct, 80  
mqc\_matrixmatrixproduct, 80  
mqc\_matrixmatrixsubtract, 80  
mqc\_matrixmatrixsum, 81  
mqc\_matrixscalarproduct, 82  
mqc\_matrixvectordotproduct, 82  
mqc\_outer, 82  
mqc\_output\_complex\_scalar, 83  
mqc\_output\_integer\_scalar, 84  
mqc\_output\_mqcscalar\_scalar, 84  
mqc\_output\_real\_scalar, 85  
mqc\_print\_matrix\_algebra1, 86  
mqc\_print\_r4tensor\_algebra1, 87  
mqc\_print\_scalar\_algebra1, 87  
mqc\_print\_vector\_algebra1, 88  
mqc\_r4tensor\_at, 89  
mqc\_r4tensor\_havecomplex, 89  
mqc\_r4tensor\_haveinteger, 89  
mqc\_r4tensor\_havereal, 89  
mqc\_r4tensor\_initialize, 89  
mqc\_r4tensor\_put, 90  
mqc\_realgtscalar, 90  
mqc\_realltscalar, 91  
mqc\_realscalaradd, 92  
mqc\_realscalardivide, 93  
mqc\_realscalarmultiply, 94  
mqc\_realscalarsubtract, 94  
mqc\_realvectorproduct, 95  
mqc\_scalar\_acos, 96  
mqc\_scalar\_asin, 96  
mqc\_scalar\_atan, 97  
mqc\_scalar\_atan2, 97  
mqc\_scalar\_cmplx, 98  
mqc\_scalar\_complex\_conjugate, 99  
mqc\_scalar\_complex\_imagpart, 99  
mqc\_scalar\_complex\_realpart, 100  
mqc\_scalar\_cos, 101  
mqc\_scalar\_get\_abs\_value, 101  
mqc\_scalar\_get\_intrinsic\_complex, 102  
mqc\_scalar\_get\_intrinsic\_integer, 102  
mqc\_scalar\_get\_intrinsic\_real, 103  
mqc\_scalar\_get\_random\_value, 104  
mqc\_scalar\_havecomplex, 105  
mqc\_scalar\_haveinteger, 105  
mqc\_scalar\_havereal, 106  
mqc\_scalar\_isallocated, 107  
mqc\_scalar\_sin, 107  
mqc\_scalar\_sqrt, 108  
mqc\_scalar\_tan, 108  
mqc\_scalaradd, 109  
mqc\_scalarcomplexadd, 110  
mqc\_scalarcomplexdivide, 110  
mqc\_scalarcomplexexponent, 111  
mqc\_scalarcomplexmultiply, 112  
mqc\_scalarcomplexsubtract, 113  
mqc\_scalardivide, 113  
mqc\_scalareq, 114  
mqc\_scalarexponent, 115  
mqc\_scalarge, 115  
mqc\_scalargt, 116  
mqc\_scalargtinteger, 117  
mqc\_scalargtreal, 118  
mqc\_scalarintegeradd, 118  
mqc\_scalarintegerdivide, 119  
mqc\_scalarintegerexponent, 120  
mqc\_scalarintegermultiply, 121  
mqc\_scalarintegersubtract, 121  
mqc\_scalarle, 122  
mqc\_scalarleinteger, 123  
mqc\_scalarlereal, 123  
mqc\_scalarlt, 124  
mqc\_scalarltreal, 125  
mqc\_scalarmatrixproduct, 126  
mqc\_scalarmultiply, 126  
mqc\_scalarne, 127  
mqc\_scalarrealadd, 127  
mqc\_scalarrealddivide, 128  
mqc\_scalarrealexponent, 129  
mqc\_scalarrealmultiply, 129  
mqc\_scalarrealsubtract, 130  
mqc\_scalarsubtract, 131  
mqc\_scalarvectordifference, 131  
mqc\_scalarvectorproduct, 132  
mqc\_scalarvectorsum, 133  
mqc\_set\_array2tensor, 133  
mqc\_set\_array2vector\_complex, 134  
mqc\_set\_array2vector\_integer, 134  
mqc\_set\_array2vector\_real, 135  
mqc\_set\_complexarray2matrix, 136  
mqc\_set\_integerarray2matrix, 137  
mqc\_set\_matrix2complexarray, 137  
mqc\_set\_matrix2integerarray, 138  
mqc\_set\_matrix2matrix, 139  
mqc\_set\_matrix2realarray, 140  
mqc\_set\_realarray2matrix, 140  
mqc\_set\_vector2complexarray, 141  
mqc\_set\_vector2integerarray, 142  
mqc\_set\_vector2realarray, 143  
mqc\_set\_vector2vector, 143  
mqc\_vector2diagmatrix, 144  
mqc\_vector\_abs, 145  
mqc\_vector\_argsort, 145  
mqc\_vector\_cast\_complex, 146  
mqc\_vector\_cast\_integer, 147  
mqc\_vector\_cast\_real, 147  
mqc\_vector\_cmplx, 148

- mqc\_vector\_complex\_imagpart, [149](#)
- mqc\_vector\_complex\_realpart, [149](#)
- mqc\_vector\_conjugate\_transpose, [150](#)
- mqc\_vector\_copy\_complex2int, [150](#)
- mqc\_vector\_copy\_complex2real, [151](#)
- mqc\_vector\_copy\_int2complex, [152](#)
- mqc\_vector\_copy\_int2real, [152](#)
- mqc\_vector\_copy\_real2complex, [153](#)
- mqc\_vector\_copy\_real2int, [153](#)
- mqc\_vector\_havecomplex, [154](#)
- mqc\_vector\_haveinteger, [155](#)
- mqc\_vector\_havereal, [155](#)
- mqc\_vector\_initialize, [157](#)
- mqc\_vector\_isallocated, [158](#)
- mqc\_vector\_iscolumn, [158](#)
- mqc\_vector\_maxloc, [159](#)
- mqc\_vector\_maxval, [159](#)
- mqc\_vector\_minloc, [160](#)
- mqc\_vector\_minval, [161](#)
- mqc\_vector\_norm, [161](#)
- mqc\_vector\_pop, [162](#)
- mqc\_vector\_power, [163](#)
- mqc\_vector\_push, [164](#)
- mqc\_vector\_scalar\_at, [164](#)
- mqc\_vector\_scalar\_increment, [165](#)
- mqc\_vector\_scalar\_put, [166](#)
- mqc\_vector\_shift, [166](#)
- mqc\_vector\_sort, [167](#)
- mqc\_vector\_sqrt, [168](#)
- mqc\_vector\_transpose, [168](#)
- mqc\_vector\_unshift, [169](#)
- mqc\_vector\_vector\_at, [170](#)
- mqc\_vector\_vector\_put, [170](#)
- mqc\_vectorcomplexdivide, [171](#)
- mqc\_vectorcomplexproduct, [172](#)
- mqc\_vectorintegerdivide, [173](#)
- mqc\_vectorintegerproduct, [173](#)
- mqc\_vectormatrixdotproduct, [174](#)
- mqc\_vectorrealddivide, [174](#)
- mqc\_vectorrealproduct, [176](#)
- mqc\_vectorscalardivide, [177](#)
- mqc\_vectorscalarproduct, [177](#)
- mqc\_vectorvectordifference, [178](#)
- mqc\_vectorvectordotproduct, [179](#)
- mqc\_vectorvectorsum, [180](#)
- symindexhash, [180](#)
- mqc\_algebra::abs, [199](#)
  - mqc\_scalar\_get\_abs\_value, [199](#)
  - mqc\_vector\_abs, [200](#)
- mqc\_algebra::acos, [201](#)
  - mqc\_scalar\_acos, [201](#)
- mqc\_algebra::aimag, [202](#)
  - mqc\_scalar\_complex\_imagpart, [202](#)
  - mqc\_vector\_complex\_imagpart, [203](#)
- mqc\_algebra::asin, [204](#)
  - mqc\_scalar\_asin, [204](#)
- mqc\_algebra::assignment(=), [205](#)
  - mqc\_input\_complex\_scalar, [206](#)
  - mqc\_input\_integer\_scalar, [207](#)
  - mqc\_input\_real\_scalar, [207](#)
  - mqc\_output\_complex\_scalar, [208](#)
  - mqc\_output\_integer\_scalar, [209](#)
  - mqc\_output\_mqcscalar\_scalar, [209](#)
  - mqc\_output\_real\_scalar, [210](#)
  - mqc\_set\_array2tensor, [211](#)
  - mqc\_set\_array2vector\_complex, [211](#)
  - mqc\_set\_array2vector\_integer, [212](#)
  - mqc\_set\_array2vector\_real, [212](#)
  - mqc\_set\_complexarray2matrix, [213](#)
  - mqc\_set\_integerarray2matrix, [214](#)
  - mqc\_set\_matrix2complexarray, [214](#)
  - mqc\_set\_matrix2integerarray, [215](#)
  - mqc\_set\_matrix2matrix, [216](#)
  - mqc\_set\_matrix2realarray, [217](#)
  - mqc\_set\_realarray2matrix, [217](#)
  - mqc\_set\_vector2complexarray, [218](#)
  - mqc\_set\_vector2integerarray, [219](#)
  - mqc\_set\_vector2realarray, [220](#)
  - mqc\_set\_vector2vector, [220](#)
- mqc\_algebra::atan, [222](#)
  - mqc\_scalar\_atan, [222](#)
- mqc\_algebra::atan2, [223](#)
  - mqc\_scalar\_atan2, [223](#)
- mqc\_algebra::cmplx, [224](#)
  - mqc\_scalar\_cmplx, [224](#)
  - mqc\_vector\_cmplx, [225](#)
- mqc\_algebra::conjg, [226](#)
  - mqc\_scalar\_complex\_conjugate, [226](#)
- mqc\_algebra::contraction, [227](#)
  - mqc\_matrix\_matrix\_contraction, [227](#)
- mqc\_algebra::cos, [228](#)
  - mqc\_scalar\_cos, [229](#)
- mqc\_algebra::dagger, [229](#)
  - mqc\_matrix\_conjugate\_transpose, [230](#)
  - mqc\_vector\_conjugate\_transpose, [230](#)
- mqc\_algebra::dot\_product, [231](#)
  - mqc\_vectorvectordotproduct, [232](#)
- mqc\_algebra::matmul, [233](#)
  - mqc\_matrixmatrixdotproduct, [234](#)
  - mqc\_matrixvectordotproduct, [234](#)
  - mqc\_vectormatrixdotproduct, [234](#)
- mqc\_algebra::matrix\_symm2sq, [236](#)
  - matrix\_symm2sq\_complex, [236](#)
  - matrix\_symm2sq\_integer, [237](#)
  - matrix\_symm2sq\_real, [238](#)
- mqc\_algebra::mqc\_cast\_complex, [239](#)
  - mqc\_matrix\_cast\_complex, [239](#)
  - mqc\_vector\_cast\_complex, [240](#)

- mqc\_algebra::mqc\_cast\_integer, [241](#)
  - mqc\_matrix\_cast\_integer, [241](#)
  - mqc\_vector\_cast\_integer, [242](#)
- mqc\_algebra::mqc\_cast\_real, [242](#)
  - mqc\_matrix\_cast\_real, [243](#)
  - mqc\_vector\_cast\_real, [243](#)
- mqc\_algebra::mqc\_have\_complex, [246](#)
  - mqc\_matrix\_havecomplex, [246](#)
  - mqc\_vector\_havecomplex, [247](#)
- mqc\_algebra::mqc\_have\_int, [247](#)
  - mqc\_matrix\_haveinteger, [248](#)
  - mqc\_vector\_haveinteger, [248](#)
- mqc\_algebra::mqc\_have\_real, [249](#)
  - mqc\_matrix\_havereal, [250](#)
  - mqc\_vector\_havereal, [250](#)
- mqc\_algebra::mqc\_matrix, [251](#)
  - at, [252](#)
  - dagger, [252](#)
  - det, [252](#)
  - diag, [253](#)
  - eigensys, [253](#)
  - identity, [253](#)
  - init, [253](#)
  - inv, [253](#)
  - mat, [253](#)
  - mput, [254](#)
  - norm, [254](#)
  - print, [254](#)
  - put, [254](#)
  - rmsmax, [254](#)
  - set, [254](#)
  - sqr, [255](#)
  - svd, [255](#)
  - trace, [255](#)
  - transpose, [255](#)
  - vat, [255](#)
  - vput, [255](#)
- mqc\_algebra::mqc\_matrix\_diagmatrix\_put, [256](#)
  - mqc\_matrix\_diagmatrix\_put\_complex, [256](#)
  - mqc\_matrix\_diagmatrix\_put\_integer, [257](#)
  - mqc\_matrix\_diagmatrix\_put\_real, [258](#)
  - mqc\_matrix\_diagmatrix\_put\_vector, [258](#)
- mqc\_algebra::mqc\_matrix\_symmmatrix\_put, [259](#)
  - mqc\_matrix\_symmmatrix\_put\_complex, [260](#)
  - mqc\_matrix\_symmmatrix\_put\_integer, [260](#)
  - mqc\_matrix\_symmmatrix\_put\_real, [261](#)
- mqc\_algebra::mqc\_print, [263](#)
  - mqc\_print\_matrix\_algebra1, [263](#)
  - mqc\_print\_r4tensor\_algebra1, [264](#)
  - mqc\_print\_scalar\_algebra1, [264](#)
  - mqc\_print\_vector\_algebra1, [265](#)
- mqc\_algebra::mqc\_r4tensor, [269](#)
  - at, [270](#)
  - init, [270](#)
  - print, [270](#)
  - put, [270](#)
- mqc\_algebra::mqc\_scalar, [271](#)
  - abs, [271](#)
  - cval, [271](#)
  - ival, [272](#)
  - print, [272](#)
  - random, [272](#)
  - rval, [272](#)
- mqc\_algebra::mqc\_set\_array2vector, [277](#)
  - mqc\_set\_array2vector\_complex, [277](#)
  - mqc\_set\_array2vector\_integer, [278](#)
  - mqc\_set\_array2vector\_real, [279](#)
- mqc\_algebra::mqc\_vector, [280](#)
  - abs, [281](#)
  - argsort, [281](#)
  - at, [282](#)
  - dagger, [282](#)
  - diag, [282](#)
  - init, [282](#)
  - maxloc, [282](#)
  - maxval, [282](#)
  - minloc, [283](#)
  - minval, [283](#)
  - norm, [283](#)
  - pop, [283](#)
  - power, [283](#)
  - print, [283](#)
  - push, [284](#)
  - put, [284](#)
  - shift, [284](#)
  - size, [284](#)
  - sort, [284](#)
  - sqr, [284](#)
  - transpose, [285](#)
  - unshift, [285](#)
  - vat, [285](#)
  - vput, [285](#)
- mqc\_algebra::operator(\*\*), [303](#)
  - mqc\_scalarcomplexexponent, [303](#)
  - mqc\_scalarexponent, [304](#)
  - mqc\_scalarintegerexponent, [305](#)
  - mqc\_scalarrealexponent, [306](#)
- mqc\_algebra::operator(\*), [289](#)
  - mqc\_complexscalarmultiply, [291](#)
  - mqc\_complexvectorproduct, [291](#)
  - mqc\_integerscalarmultiply, [292](#)
  - mqc\_integervectorproduct, [293](#)
  - mqc\_matrixmatrixproduct, [293](#)
  - mqc\_matrixscalarproduct, [294](#)
  - mqc\_realscalarmultiply, [294](#)
  - mqc\_realvectorproduct, [295](#)
  - mqc\_scalarcomplexmultiply, [296](#)
  - mqc\_scalarintegermultiply, [296](#)



- mqc\_scalarmatrixproduct, 297
- mqc\_scalarmultiply, 297
- mqc\_scalarrealmultiply, 298
- mqc\_scalarvectorproduct, 299
- mqc\_vectorcomplexproduct, 299
- mqc\_vectorintegerproduct, 300
- mqc\_vectorrealproduct, 301
- mqc\_vectorscalarproduct, 301
- mqc\_algebra::operator(+), 306
  - mqc\_complexscalaradd, 307
  - mqc\_integerscalaradd, 308
  - mqc\_matrixmatrixsum, 309
  - mqc\_realscalaradd, 309
  - mqc\_scalaradd, 310
  - mqc\_scalarcomplexadd, 311
  - mqc\_scalarintegeradd, 311
  - mqc\_scalarrealadd, 312
  - mqc\_scalarvectorsum, 313
  - mqc\_vectorvectorsum, 313
- mqc\_algebra::operator(-), 315
  - mqc\_complexscalarsubtract, 316
  - mqc\_integerscalarsubtract, 316
  - mqc\_matrixmatrixsubtract, 317
  - mqc\_realscalarsubtract, 318
  - mqc\_scalarcomplexsubtract, 318
  - mqc\_scalarintegersubtract, 319
  - mqc\_scalarrealsubtract, 320
  - mqc\_scalarsubtract, 320
  - mqc\_scalarvectordifference, 321
  - mqc\_vectorvectordifference, 322
- mqc\_algebra::operator(.dot.), 323
  - mqc\_matrixmatrixdotproduct, 323
  - mqc\_matrixvectordotproduct, 323
  - mqc\_vectormatrixdotproduct, 324
  - mqc\_vectorvectordotproduct, 324
- mqc\_algebra::operator(.eq.), 325
  - mqc\_scalareq, 325
- mqc\_algebra::operator(.ewd.), 326
  - mqc\_elementmatrixdivide, 326
- mqc\_algebra::operator(.ewp.), 327
  - mqc\_elementmatrixproduct, 327
  - mqc\_elementvectorproduct, 328
- mqc\_algebra::operator(.ge.), 329
  - mqc\_scalarge, 329
- mqc\_algebra::operator(.gt.), 330
  - mqc\_integertscalar, 331
  - mqc\_realgtscalar, 332
  - mqc\_scalargt, 332
  - mqc\_scalargtinteger, 333
  - mqc\_scalargtreal, 334
- mqc\_algebra::operator(.le.), 335
  - mqc\_integerlescalar, 336
  - mqc\_reallescalar, 336
  - mqc\_scalarle, 337
  - mqc\_scalarleinteger, 338
  - mqc\_scalarlereal, 339
- mqc\_algebra::operator(.lt.), 340
  - mqc\_realltscalar, 340
  - mqc\_scalarlt, 341
  - mqc\_scalarltreal, 341
- mqc\_algebra::operator(.ne.), 342
  - mqc\_scalarne, 343
- mqc\_algebra::operator(.outer.), 344
  - mqc\_outer, 344
- mqc\_algebra::operator(.x.), 345
  - mqc\_crossproduct, 345
- mqc\_algebra::operator(/), 346
  - mqc\_complexscalardivide, 347
  - mqc\_integerscalardivide, 347
  - mqc\_realscalardivide, 348
  - mqc\_scalarcomplexdivide, 349
  - mqc\_scalardivide, 349
  - mqc\_scalarintegerdivide, 350
  - mqc\_scalarrealddivide, 351
  - mqc\_vectorcomplexdivide, 352
  - mqc\_vectorintegerdivide, 352
  - mqc\_vectorrealddivide, 353
  - mqc\_vectorscalardivide, 354
- mqc\_algebra::real, 354
  - mqc\_scalar\_complex\_realpart, 355
  - mqc\_vector\_complex\_realpart, 355
- mqc\_algebra::sin, 356
  - mqc\_scalar\_sin, 357
- mqc\_algebra::sqrt, 357
  - mqc\_scalar\_sqrt, 358
- mqc\_algebra::tan, 358
  - mqc\_scalar\_tan, 359
- mqc\_algebra::transpose, 360
  - mqc\_matrix\_transpose, 360
  - mqc\_vector\_transpose, 361
- mqc\_allocate\_matrix
  - mqc\_algebra, 28
- mqc\_allocate\_r4tensor
  - mqc\_algebra, 29
- mqc\_allocate\_scalar
  - mqc\_algebra, 29
- mqc\_allocate\_vector
  - mqc\_algebra, 30
- mqc\_build\_ci\_hamiltonian
  - mqc\_est, 184
- mqc\_complexscalaradd
  - mqc\_algebra, 31
  - mqc\_algebra::operator(+), 307
- mqc\_complexscalardivide
  - mqc\_algebra, 32
  - mqc\_algebra::operator(/), 347
- mqc\_complexscalarmultiply
  - mqc\_algebra, 33



- mqc\_algebra::operator(\*), 291
- mqc\_complexscalarsubtract
  - mqc\_algebra, 33
  - mqc\_algebra::operator(-), 316
- mqc\_complexvectorproduct
  - mqc\_algebra, 34
  - mqc\_algebra::operator(\*), 291
- mqc\_crossproduct
  - mqc\_algebra, 35
  - mqc\_algebra::operator(.x.), 345
- mqc\_deallocate\_matrix
  - mqc\_algebra, 35
- mqc\_deallocate\_r4tensor
  - mqc\_algebra, 36
- mqc\_deallocate\_scalar
  - mqc\_algebra, 36
- mqc\_deallocate\_vector
  - mqc\_algebra, 37
- mqc\_eigenvalue\_eigenvalue\_dotproduct
  - mqc\_est, 184
  - mqc\_est::dot\_product, 233
- mqc\_eigenvalues\_add\_name
  - mqc\_est, 184
- mqc\_eigenvalues\_allocate
  - mqc\_est, 185
- mqc\_eigenvalues\_array\_name
  - mqc\_est, 185
- mqc\_eigenvalues\_array\_type
  - mqc\_est, 185
- mqc\_eigenvalues\_at
  - mqc\_est, 185
- mqc\_eigenvalues\_dimension
  - mqc\_est, 185
- mqc\_eigenvalues\_eigenvalues\_multiply
  - mqc\_est, 186
  - mqc\_est::matmul, 234
- mqc\_eigenvalues\_has\_alpha
  - mqc\_est, 186
- mqc\_eigenvalues\_has\_beta
  - mqc\_est, 186
- mqc\_eigenvalues\_integral\_multiply
  - mqc\_est, 186
  - mqc\_est::matmul, 235
- mqc\_eigenvalues\_isallocated
  - mqc\_est, 186
- mqc\_eigenvalues\_output\_array
  - mqc\_est, 186
  - mqc\_est::assignment(=), 222
- mqc\_eigenvalues\_output\_block
  - mqc\_est, 187
- mqc\_elementmatrixdivide
  - mqc\_algebra, 37
  - mqc\_algebra::operator(.ewd.), 326
- mqc\_elementmatrixproduct
  - mqc\_algebra, 38
  - mqc\_algebra::operator(.ewp.), 327
- mqc\_elementvectorproduct
  - mqc\_algebra, 39
  - mqc\_algebra::operator(.ewp.), 328
- mqc\_eri\_integral\_contraction
  - mqc\_est, 187
  - mqc\_est::contraction, 228
- mqc\_est, 181
  - gen\_det\_str, 183
  - get\_one\_gamma\_matrix, 184
  - mqc\_build\_ci\_hamiltonian, 184
  - mqc\_eigenvalue\_eigenvalue\_dotproduct, 184
  - mqc\_eigenvalues\_add\_name, 184
  - mqc\_eigenvalues\_allocate, 185
  - mqc\_eigenvalues\_array\_name, 185
  - mqc\_eigenvalues\_array\_type, 185
  - mqc\_eigenvalues\_at, 185
  - mqc\_eigenvalues\_dimension, 185
  - mqc\_eigenvalues\_eigenvalues\_multiply, 186
  - mqc\_eigenvalues\_has\_alpha, 186
  - mqc\_eigenvalues\_has\_beta, 186
  - mqc\_eigenvalues\_integral\_multiply, 186
  - mqc\_eigenvalues\_isallocated, 186
  - mqc\_eigenvalues\_output\_array, 186
  - mqc\_eigenvalues\_output\_block, 187
  - mqc\_eri\_integral\_contraction, 187
  - mqc\_integral\_add\_name, 187
  - mqc\_integral\_allocate, 187
  - mqc\_integral\_array\_name, 187
  - mqc\_integral\_array\_type, 188
  - mqc\_integral\_at, 188
  - mqc\_integral\_conjugate\_transpose, 188
  - mqc\_integral\_delete\_energy\_list, 188
  - mqc\_integral\_difference, 188
  - mqc\_integral\_dimension, 188
  - mqc\_integral\_eigenvalues\_multiply, 189
  - mqc\_integral\_get\_energy\_list, 189
  - mqc\_integral\_has\_alpha, 189
  - mqc\_integral\_has\_alphabeta, 189
  - mqc\_integral\_has\_beta, 189
  - mqc\_integral\_has\_betaalpha, 189
  - mqc\_integral\_identity, 190
  - mqc\_integral\_initialize, 190
  - mqc\_integral\_integral\_multiply, 190
  - mqc\_integral\_isallocated, 190
  - mqc\_integral\_matrix\_multiply, 190
  - mqc\_integral\_norm, 191
  - mqc\_integral\_output\_array, 191
  - mqc\_integral\_output\_block, 191
  - mqc\_integral\_output\_orbitals, 191
  - mqc\_integral\_scalar\_multiply, 191
  - mqc\_integral\_set\_energy\_list, 192
  - mqc\_integral\_sum, 192

- mqc\_integral\_swap\_orbitals, 192
- mqc\_integral\_transpose, 192
- mqc\_matrix\_integral\_multiply, 192
- mqc\_matrix\_spinblockghf, 193
- mqc\_matrix\_undospinblockghf\_eigenvalues, 193
- mqc\_matrix\_undospinblockghf\_integral, 193
- mqc\_print\_eigenvalues, 193
- mqc\_print\_integral, 193
- mqc\_print\_twoeris, 194
- mqc\_print\_wavefunction, 194
- mqc\_scalar\_integral\_multiply, 194
- mqc\_scf\_eigenvalues\_power, 194
- mqc\_scf\_integral\_contraction, 194
- mqc\_scf\_integral\_determinant, 195
- mqc\_scf\_integral\_diagonalize, 195
- mqc\_scf\_integral\_generalized\_eigensystem, 195
- mqc\_scf\_integral\_inverse, 195
- mqc\_scf\_integral\_trace, 195
- mqc\_scf\_transformation\_matrix, 195
- mqc\_twoeris\_allocate, 196
- mqc\_twoeris\_at, 196
- slater\_condon, 196
- twoeri\_trans, 196
- mqc\_est::assignment(=), 221
  - mqc\_eigenvalues\_output\_array, 222
  - mqc\_integral\_output\_array, 222
- mqc\_est::contraction, 228
  - mqc\_eri\_integral\_contraction, 228
  - mqc\_scf\_integral\_contraction, 228
- mqc\_est::dagger, 231
  - mqc\_integral\_conjugate\_transpose, 231
- mqc\_est::dot\_product, 233
  - mqc\_eigenvalue\_eigenvalue\_dotproduct, 233
- mqc\_est::matmul, 234
  - mqc\_eigenvalues\_eigenvalues\_multiply, 234
  - mqc\_eigenvalues\_integral\_multiply, 235
  - mqc\_integral\_eigenvalues\_multiply, 235
  - mqc\_integral\_integral\_multiply, 235
  - mqc\_integral\_matrix\_multiply, 235
  - mqc\_matrix\_integral\_multiply, 235
- mqc\_est::mqc\_determinant, 244
  - nalpstr, 244
  - nbetstr, 244
  - ndets, 244
  - order, 245
  - strings, 245
- mqc\_est::mqc\_determinant\_string, 245
  - alpha, 245
  - beta, 245
- mqc\_est::mqc\_matrix\_undospinblockghf, 262
  - mqc\_matrix\_undospinblockghf\_eigenvalues, 262
  - mqc\_matrix\_undospinblockghf\_integral, 262
- mqc\_est::mqc\_print, 267
  - mqc\_print\_eigenvalues, 267
  - mqc\_print\_integral, 267
  - mqc\_print\_twoeris, 267
  - mqc\_print\_wavefunction, 268
- mqc\_est::mqc\_pscf\_wavefunction, 268
  - nactive, 268
  - ncore, 269
  - nfrz, 269
  - nval, 269
  - pscf\_amplitudes, 269
  - pscf\_energies, 269
- mqc\_est::mqc\_scf\_eigenvalues, 273
  - addlabel, 273
  - at, 273
  - getblock, 273
  - getlabel, 273
  - power, 273
  - print, 273
- mqc\_est::mqc\_scf\_integral, 274
  - addlabel, 274
  - deleteelist, 274
  - det, 274
  - diag, 275
  - eigensys, 275
  - getblock, 275
  - getelist, 275
  - getlabel, 275
  - identity, 275
  - init, 275
  - inv, 276
  - norm, 276
  - orbitals, 276
  - print, 276
  - setelist, 276
  - swap, 276
  - trace, 276
- mqc\_est::mqc\_twoeris, 280
  - print, 280
- mqc\_est::mqc\_wavefunction, 286
  - basis, 287
  - charge, 287
  - core\_hamiltonian, 287
  - density\_matrix, 287
  - fock\_matrix, 287
  - mo\_coefficients, 287
  - mo\_energies, 287
  - mo\_symmetries, 288
  - multiplicity, 288
  - nalpstr, 288
  - nbasis, 288
  - nbeta, 288
  - nelectrons, 288
  - overlap\_matrix, 288
  - print, 286
  - scf\_density\_matrix, 289

- symmetry, 289
- wf\_complex, 289
- wf\_type, 289
- mqc\_est::operator(\*), 302
  - mqc\_integral\_scalar\_multiply, 302
  - mqc\_scalar\_integral\_multiply, 303
- mqc\_est::operator(+), 314
  - mqc\_integral\_sum, 314
- mqc\_est::operator(-), 315
  - mqc\_integral\_difference, 315
- mqc\_est::transpose, 360
  - mqc\_integral\_transpose, 360
- mqc\_givens\_matrix
  - mqc\_algebra, 40
- mqc\_input\_complex\_scalar
  - mqc\_algebra, 40
  - mqc\_algebra::assignment(=), 206
- mqc\_input\_integer\_scalar
  - mqc\_algebra, 41
  - mqc\_algebra::assignment(=), 207
- mqc\_input\_real\_scalar
  - mqc\_algebra, 41
  - mqc\_algebra::assignment(=), 207
- mqc\_integergtscalar
  - mqc\_algebra, 42
  - mqc\_algebra::operator(.gt.), 331
- mqc\_integerlescalar
  - mqc\_algebra, 43
  - mqc\_algebra::operator(.le.), 336
- mqc\_integerscalaradd
  - mqc\_algebra, 43
  - mqc\_algebra::operator(+), 308
- mqc\_integerscalardivide
  - mqc\_algebra, 44
  - mqc\_algebra::operator(/), 347
- mqc\_integerscalarmultiply
  - mqc\_algebra, 45
  - mqc\_algebra::operator(\*), 292
- mqc\_integerscalarsubtract
  - mqc\_algebra, 46
  - mqc\_algebra::operator(-), 316
- mqc\_integervectorproduct
  - mqc\_algebra, 46
  - mqc\_algebra::operator(\*), 293
- mqc\_integral\_add\_name
  - mqc\_est, 187
- mqc\_integral\_allocate
  - mqc\_est, 187
- mqc\_integral\_array\_name
  - mqc\_est, 187
- mqc\_integral\_array\_type
  - mqc\_est, 188
- mqc\_integral\_at
  - mqc\_est, 188
- mqc\_integral\_conjugate\_transpose
  - mqc\_est, 188
  - mqc\_est::dagger, 231
- mqc\_integral\_delete\_energy\_list
  - mqc\_est, 188
- mqc\_integral\_difference
  - mqc\_est, 188
  - mqc\_est::operator(-), 315
- mqc\_integral\_dimension
  - mqc\_est, 188
- mqc\_integral\_eigenvalues\_multiply
  - mqc\_est, 189
  - mqc\_est::matmul, 235
- mqc\_integral\_get\_energy\_list
  - mqc\_est, 189
- mqc\_integral\_has\_alpha
  - mqc\_est, 189
- mqc\_integral\_has\_alphabeta
  - mqc\_est, 189
- mqc\_integral\_has\_beta
  - mqc\_est, 189
- mqc\_integral\_has\_betaalpha
  - mqc\_est, 189
- mqc\_integral\_identity
  - mqc\_est, 190
- mqc\_integral\_initialize
  - mqc\_est, 190
- mqc\_integral\_integral\_multiply
  - mqc\_est, 190
  - mqc\_est::matmul, 235
- mqc\_integral\_isallocated
  - mqc\_est, 190
- mqc\_integral\_matrix\_multiply
  - mqc\_est, 190
  - mqc\_est::matmul, 235
- mqc\_integral\_norm
  - mqc\_est, 191
- mqc\_integral\_output\_array
  - mqc\_est, 191
  - mqc\_est::assignment(=), 222
- mqc\_integral\_output\_block
  - mqc\_est, 191
- mqc\_integral\_output\_orbitals
  - mqc\_est, 191
- mqc\_integral\_scalar\_multiply
  - mqc\_est, 191
  - mqc\_est::operator(\*), 302
- mqc\_integral\_set\_energy\_list
  - mqc\_est, 192
- mqc\_integral\_sum
  - mqc\_est, 192
  - mqc\_est::operator(+), 314
- mqc\_integral\_swap\_orbitals
  - mqc\_est, 192

- mqc\_integral\_transpose
  - mqc\_est, [192](#)
  - mqc\_est::transpose, [360](#)
- mqc\_length\_vector
  - mqc\_algebra, [47](#)
- mqc\_matrix\_cast\_complex
  - mqc\_algebra, [47](#)
  - mqc\_algebra::mqc\_cast\_complex, [239](#)
- mqc\_matrix\_cast\_integer
  - mqc\_algebra, [48](#)
  - mqc\_algebra::mqc\_cast\_integer, [241](#)
- mqc\_matrix\_cast\_real
  - mqc\_algebra, [49](#)
  - mqc\_algebra::mqc\_cast\_real, [243](#)
- mqc\_matrix\_columns
  - mqc\_algebra, [49](#)
- mqc\_matrix\_conjugate\_transpose
  - mqc\_algebra, [50](#)
  - mqc\_algebra::dagger, [230](#)
- mqc\_matrix\_copy\_complex2int
  - mqc\_algebra, [50](#)
- mqc\_matrix\_copy\_complex2real
  - mqc\_algebra, [51](#)
- mqc\_matrix\_copy\_int2complex
  - mqc\_algebra, [52](#)
- mqc\_matrix\_copy\_int2real
  - mqc\_algebra, [52](#)
- mqc\_matrix\_copy\_real2complex
  - mqc\_algebra, [53](#)
- mqc\_matrix\_copy\_real2int
  - mqc\_algebra, [53](#)
- mqc\_matrix\_determinant
  - mqc\_algebra, [54](#)
- mqc\_matrix\_diag2full
  - mqc\_algebra, [54](#)
- mqc\_matrix\_diag2symm
  - mqc\_algebra, [55](#)
- mqc\_matrix\_diagmatrix\_put\_complex
  - mqc\_algebra, [55](#)
  - mqc\_algebra::mqc\_matrix\_diagmatrix\_put, [256](#)
- mqc\_matrix\_diagmatrix\_put\_integer
  - mqc\_algebra, [56](#)
  - mqc\_algebra::mqc\_matrix\_diagmatrix\_put, [257](#)
- mqc\_matrix\_diagmatrix\_put\_real
  - mqc\_algebra, [57](#)
  - mqc\_algebra::mqc\_matrix\_diagmatrix\_put, [258](#)
- mqc\_matrix\_diagmatrix\_put\_vector
  - mqc\_algebra, [58](#)
  - mqc\_algebra::mqc\_matrix\_diagmatrix\_put, [258](#)
- mqc\_matrix\_diagonalize
  - mqc\_algebra, [58](#)
- mqc\_matrix\_full2diag
  - mqc\_algebra, [59](#)
- mqc\_matrix\_full2symm
  - mqc\_algebra, [60](#)
- mqc\_matrix\_generalized\_eigensystem
  - mqc\_algebra, [60](#)
- mqc\_matrix\_havecomplex
  - mqc\_algebra, [61](#)
  - mqc\_algebra::mqc\_have\_complex, [246](#)
- mqc\_matrix\_havediagonal
  - mqc\_algebra, [61](#)
- mqc\_matrix\_havefull
  - mqc\_algebra, [62](#)
- mqc\_matrix\_haveinteger
  - mqc\_algebra, [62](#)
  - mqc\_algebra::mqc\_have\_int, [248](#)
- mqc\_matrix\_havereal
  - mqc\_algebra, [63](#)
  - mqc\_algebra::mqc\_have\_real, [250](#)
- mqc\_matrix\_havesymmetric
  - mqc\_algebra, [64](#)
- mqc\_matrix\_identity
  - mqc\_algebra, [64](#)
- mqc\_matrix\_initialize
  - mqc\_algebra, [64](#)
- mqc\_matrix\_integral\_multiply
  - mqc\_est, [192](#)
  - mqc\_est::matmul, [235](#)
- mqc\_matrix\_inverse
  - mqc\_algebra, [65](#)
- mqc\_matrix\_isallocated
  - mqc\_algebra, [65](#)
- mqc\_matrix\_matrix\_at
  - mqc\_algebra, [65](#)
- mqc\_matrix\_matrix\_contraction
  - mqc\_algebra, [66](#)
  - mqc\_algebra::contraction, [227](#)
- mqc\_matrix\_matrix\_put
  - mqc\_algebra, [66](#)
- mqc\_matrix\_norm
  - mqc\_algebra, [68](#)
- mqc\_matrix\_rms\_max
  - mqc\_algebra, [68](#)
- mqc\_matrix\_rows
  - mqc\_algebra, [68](#)
- mqc\_matrix\_scalar\_at
  - mqc\_algebra, [69](#)
- mqc\_matrix\_scalar\_put
  - mqc\_algebra, [69](#)
- mqc\_matrix\_set
  - mqc\_algebra, [70](#)
- mqc\_matrix\_spinblockghf
  - mqc\_est, [193](#)
- mqc\_matrix\_sqrt
  - mqc\_algebra, [70](#)
- mqc\_matrix\_svd
  - mqc\_algebra, [70](#)

mqc\_matrix\_symm2diag  
     mqc\_algebra, 70  
 mqc\_matrix\_symm2full  
     mqc\_algebra, 71  
 mqc\_matrix\_symm2full\_func  
     mqc\_algebra, 72  
 mqc\_matrix\_symmetrize  
     mqc\_algebra, 72  
 mqc\_matrix\_symmmatrix\_put\_complex  
     mqc\_algebra, 73  
     mqc\_algebra::mqc\_matrix\_symmmatrix\_put, 260  
 mqc\_matrix\_symmmatrix\_put\_integer  
     mqc\_algebra, 74  
     mqc\_algebra::mqc\_matrix\_symmmatrix\_put, 260  
 mqc\_matrix\_symmmatrix\_put\_real  
     mqc\_algebra, 74  
     mqc\_algebra::mqc\_matrix\_symmmatrix\_put, 261  
 mqc\_matrix\_symmsymmr4tensor\_put\_complex  
     mqc\_algebra, 75  
 mqc\_matrix\_symmsymmr4tensor\_put\_real  
     mqc\_algebra, 75  
 mqc\_matrix\_test\_diagonal  
     mqc\_algebra, 75  
 mqc\_matrix\_test\_symmetric  
     mqc\_algebra, 76  
 mqc\_matrix\_trace  
     mqc\_algebra, 77  
 mqc\_matrix\_transpose  
     mqc\_algebra, 77  
     mqc\_algebra::transpose, 360  
 mqc\_matrix\_undospinblockghf\_eigenvalues  
     mqc\_est, 193  
     mqc\_est::mqc\_matrix\_undospinblockghf, 262  
 mqc\_matrix\_undospinblockghf\_integral  
     mqc\_est, 193  
     mqc\_est::mqc\_matrix\_undospinblockghf, 262  
 mqc\_matrix\_vector\_at  
     mqc\_algebra, 77  
 mqc\_matrix\_vector\_put  
     mqc\_algebra, 78  
 mqc\_matrixmatrixdotproduct  
     mqc\_algebra, 80  
     mqc\_algebra::matmul, 234  
     mqc\_algebra::operator(.dot.), 323  
 mqc\_matrixmatrixproduct  
     mqc\_algebra, 80  
     mqc\_algebra::operator(\*), 293  
 mqc\_matrixmatrixsubtract  
     mqc\_algebra, 80  
     mqc\_algebra::operator(-), 317  
 mqc\_matrixmatrixsum  
     mqc\_algebra, 81  
     mqc\_algebra::operator(+), 309  
 mqc\_matrixscalarproduct  
     mqc\_algebra, 82  
     mqc\_algebra::operator(\*), 294  
 mqc\_matrixvectordotproduct  
     mqc\_algebra, 82  
     mqc\_algebra::matmul, 234  
     mqc\_algebra::operator(.dot.), 323  
 mqc\_outer  
     mqc\_algebra, 82  
     mqc\_algebra::operator(.outer.), 344  
 mqc\_output\_complex\_scalar  
     mqc\_algebra, 83  
     mqc\_algebra::assignment(=), 208  
 mqc\_output\_integer\_scalar  
     mqc\_algebra, 84  
     mqc\_algebra::assignment(=), 209  
 mqc\_output\_mqcscalar\_scalar  
     mqc\_algebra, 84  
     mqc\_algebra::assignment(=), 209  
 mqc\_output\_real\_scalar  
     mqc\_algebra, 85  
     mqc\_algebra::assignment(=), 210  
 mqc\_print\_eigenvalues  
     mqc\_est, 193  
     mqc\_est::mqc\_print, 267  
 mqc\_print\_integral  
     mqc\_est, 193  
     mqc\_est::mqc\_print, 267  
 mqc\_print\_matrix\_algebra1  
     mqc\_algebra, 86  
     mqc\_algebra::mqc\_print, 263  
 mqc\_print\_r4tensor\_algebra1  
     mqc\_algebra, 87  
     mqc\_algebra::mqc\_print, 264  
 mqc\_print\_scalar\_algebra1  
     mqc\_algebra, 87  
     mqc\_algebra::mqc\_print, 264  
 mqc\_print\_twoeris  
     mqc\_est, 194  
     mqc\_est::mqc\_print, 267  
 mqc\_print\_vector\_algebra1  
     mqc\_algebra, 88  
     mqc\_algebra::mqc\_print, 265  
 mqc\_print\_wavefunction  
     mqc\_est, 194  
     mqc\_est::mqc\_print, 268  
 mqc\_r4tensor\_at  
     mqc\_algebra, 89  
 mqc\_r4tensor\_havecomplex  
     mqc\_algebra, 89  
 mqc\_r4tensor\_haveinteger  
     mqc\_algebra, 89  
 mqc\_r4tensor\_havereal  
     mqc\_algebra, 89  
 mqc\_r4tensor\_initialize

- mqc\_algebra, 89
- mqc\_r4tensor\_put
  - mqc\_algebra, 90
- mqc\_realgtscalar
  - mqc\_algebra, 90
  - mqc\_algebra::operator(.gt.), 332
- mqc\_reallscalar
  - mqc\_algebra, 91
  - mqc\_algebra::operator(.le.), 336
- mqc\_realltscalar
  - mqc\_algebra, 91
  - mqc\_algebra::operator(.lt.), 340
- mqc\_realscalaradd
  - mqc\_algebra, 92
  - mqc\_algebra::operator(+), 309
- mqc\_realscalardivide
  - mqc\_algebra, 93
  - mqc\_algebra::operator(/), 348
- mqc\_realscalarmultiply
  - mqc\_algebra, 94
  - mqc\_algebra::operator(\*), 294
- mqc\_realscalarsubtract
  - mqc\_algebra, 94
  - mqc\_algebra::operator(-), 318
- mqc\_realvectorproduct
  - mqc\_algebra, 95
  - mqc\_algebra::operator(\*), 295
- mqc\_scalar\_acos
  - mqc\_algebra, 96
  - mqc\_algebra::acos, 201
- mqc\_scalar\_asin
  - mqc\_algebra, 96
  - mqc\_algebra::asin, 204
- mqc\_scalar\_atan
  - mqc\_algebra, 97
  - mqc\_algebra::atan, 222
- mqc\_scalar\_atan2
  - mqc\_algebra, 97
  - mqc\_algebra::atan2, 223
- mqc\_scalar\_cmplx
  - mqc\_algebra, 98
  - mqc\_algebra::cmplx, 224
- mqc\_scalar\_complex\_conjugate
  - mqc\_algebra, 99
  - mqc\_algebra::conj, 226
- mqc\_scalar\_complex\_imagpart
  - mqc\_algebra, 99
  - mqc\_algebra::aimag, 202
- mqc\_scalar\_complex\_realpart
  - mqc\_algebra, 100
  - mqc\_algebra::real, 355
- mqc\_scalar\_cos
  - mqc\_algebra, 101
  - mqc\_algebra::cos, 229
- mqc\_scalar\_get\_abs\_value
  - mqc\_algebra, 101
  - mqc\_algebra::abs, 199
- mqc\_scalar\_get\_intrinsic\_complex
  - mqc\_algebra, 102
- mqc\_scalar\_get\_intrinsic\_integer
  - mqc\_algebra, 102
- mqc\_scalar\_get\_intrinsic\_real
  - mqc\_algebra, 103
- mqc\_scalar\_get\_random\_value
  - mqc\_algebra, 104
- mqc\_scalar\_havecomplex
  - mqc\_algebra, 105
- mqc\_scalar\_haveinteger
  - mqc\_algebra, 105
- mqc\_scalar\_havereal
  - mqc\_algebra, 106
- mqc\_scalar\_integral\_multiply
  - mqc\_est, 194
  - mqc\_est::operator(\*), 303
- mqc\_scalar\_isallocated
  - mqc\_algebra, 107
- mqc\_scalar\_sin
  - mqc\_algebra, 107
  - mqc\_algebra::sin, 357
- mqc\_scalar\_sqrt
  - mqc\_algebra, 108
  - mqc\_algebra::sqrt, 358
- mqc\_scalar\_tan
  - mqc\_algebra, 108
  - mqc\_algebra::tan, 359
- mqc\_scalaradd
  - mqc\_algebra, 109
  - mqc\_algebra::operator(+), 310
- mqc\_scalarcomplexadd
  - mqc\_algebra, 110
  - mqc\_algebra::operator(+), 311
- mqc\_scalarcomplexdivide
  - mqc\_algebra, 110
  - mqc\_algebra::operator(/), 349
- mqc\_scalarcomplexexponent
  - mqc\_algebra, 111
  - mqc\_algebra::operator(\*\*), 303
- mqc\_scalarcomplexmultiply
  - mqc\_algebra, 112
  - mqc\_algebra::operator(\*), 296
- mqc\_scalarcomplexsubtract
  - mqc\_algebra, 113
  - mqc\_algebra::operator(-), 318
- mqc\_scalardivide
  - mqc\_algebra, 113
  - mqc\_algebra::operator(/), 349
- mqc\_scalareq
  - mqc\_algebra, 114

- mqc\_algebra::operator(.eq.), 325
- mqc\_scalar<sup>exponent</sup>
  - mqc\_algebra, 115
  - mqc\_algebra::operator(\*\*), 304
- mqc\_scalar<sup>large</sup>
  - mqc\_algebra, 115
  - mqc\_algebra::operator(.ge.), 329
- mqc\_scalar<sup>lgt</sup>
  - mqc\_algebra, 116
  - mqc\_algebra::operator(.gt.), 332
- mqc\_scalar<sup>lgtinteger</sup>
  - mqc\_algebra, 117
  - mqc\_algebra::operator(.gt.), 333
- mqc\_scalar<sup>lgtreal</sup>
  - mqc\_algebra, 118
  - mqc\_algebra::operator(.gt.), 334
- mqc\_scalar<sup>integeradd</sup>
  - mqc\_algebra, 118
  - mqc\_algebra::operator(+), 311
- mqc\_scalar<sup>integerdivide</sup>
  - mqc\_algebra, 119
  - mqc\_algebra::operator(/), 350
- mqc\_scalar<sup>integerexponent</sup>
  - mqc\_algebra, 120
  - mqc\_algebra::operator(\*\*), 305
- mqc\_scalar<sup>integermultiply</sup>
  - mqc\_algebra, 121
  - mqc\_algebra::operator(\*), 296
- mqc\_scalar<sup>integersubtract</sup>
  - mqc\_algebra, 121
  - mqc\_algebra::operator(-), 319
- mqc\_scalar<sup>le</sup>
  - mqc\_algebra, 122
  - mqc\_algebra::operator(.le.), 337
- mqc\_scalar<sup>leinteger</sup>
  - mqc\_algebra, 123
  - mqc\_algebra::operator(.le.), 338
- mqc\_scalar<sup>lereal</sup>
  - mqc\_algebra, 123
  - mqc\_algebra::operator(.le.), 339
- mqc\_scalar<sup>lgt</sup>
  - mqc\_algebra, 124
  - mqc\_algebra::operator(.lt.), 341
- mqc\_scalar<sup>lgtreal</sup>
  - mqc\_algebra, 125
  - mqc\_algebra::operator(.lt.), 341
- mqc\_scalar<sup>matrixproduct</sup>
  - mqc\_algebra, 126
  - mqc\_algebra::operator(\*), 297
- mqc\_scalar<sup>multiply</sup>
  - mqc\_algebra, 126
  - mqc\_algebra::operator(\*), 297
- mqc\_scalar<sup>ne</sup>
  - mqc\_algebra, 127
- mqc\_algebra::operator(.ne.), 343
- mqc\_scalar<sup>realadd</sup>
  - mqc\_algebra, 127
  - mqc\_algebra::operator(+), 312
- mqc\_scalar<sup>realddivide</sup>
  - mqc\_algebra, 128
  - mqc\_algebra::operator(/), 351
- mqc\_scalar<sup>realexponent</sup>
  - mqc\_algebra, 129
  - mqc\_algebra::operator(\*\*), 306
- mqc\_scalar<sup>realmultiply</sup>
  - mqc\_algebra, 129
  - mqc\_algebra::operator(\*), 298
- mqc\_scalar<sup>realsubtract</sup>
  - mqc\_algebra, 130
  - mqc\_algebra::operator(-), 320
- mqc\_scalar<sup>subtract</sup>
  - mqc\_algebra, 131
  - mqc\_algebra::operator(-), 320
- mqc\_scalar<sup>vectordifference</sup>
  - mqc\_algebra, 131
  - mqc\_algebra::operator(-), 321
- mqc\_scalar<sup>vectorproduct</sup>
  - mqc\_algebra, 132
  - mqc\_algebra::operator(\*), 299
- mqc\_scalar<sup>vectorsum</sup>
  - mqc\_algebra, 133
  - mqc\_algebra::operator(+), 313
- mqc\_scf\_eigenvalues<sup>power</sup>
  - mqc\_est, 194
- mqc\_scf\_integral<sup>contraction</sup>
  - mqc\_est, 194
  - mqc\_est::contraction, 228
- mqc\_scf\_integral<sup>determinant</sup>
  - mqc\_est, 195
- mqc\_scf\_integral<sup>diagonalize</sup>
  - mqc\_est, 195
- mqc\_scf\_integral<sup>generalized\_eigensystem</sup>
  - mqc\_est, 195
- mqc\_scf\_integral<sup>inverse</sup>
  - mqc\_est, 195
- mqc\_scf\_integral<sup>trace</sup>
  - mqc\_est, 195
- mqc\_scf\_transformation<sup>matrix</sup>
  - mqc\_est, 195
- mqc\_set\_array<sup>2tensor</sup>
  - mqc\_algebra, 133
  - mqc\_algebra::assignment(=), 211
- mqc\_set\_array<sup>2vector\_complex</sup>
  - mqc\_algebra, 134
  - mqc\_algebra::assignment(=), 211
  - mqc\_algebra::mqc\_set\_array2vector, 277
- mqc\_set\_array<sup>2vector\_integer</sup>
  - mqc\_algebra, 134



- mqc\_algebra::assignment(=), 212
- mqc\_algebra::mqc\_set\_array2vector, 278
- mqc\_set\_array2vector\_real
  - mqc\_algebra, 135
  - mqc\_algebra::assignment(=), 212
  - mqc\_algebra::mqc\_set\_array2vector, 279
- mqc\_set\_complexarray2matrix
  - mqc\_algebra, 136
  - mqc\_algebra::assignment(=), 213
- mqc\_set\_integerarray2matrix
  - mqc\_algebra, 137
  - mqc\_algebra::assignment(=), 214
- mqc\_set\_matrix2complexarray
  - mqc\_algebra, 137
  - mqc\_algebra::assignment(=), 214
- mqc\_set\_matrix2integerarray
  - mqc\_algebra, 138
  - mqc\_algebra::assignment(=), 215
- mqc\_set\_matrix2matrix
  - mqc\_algebra, 139
  - mqc\_algebra::assignment(=), 216
- mqc\_set\_matrix2realarray
  - mqc\_algebra, 140
  - mqc\_algebra::assignment(=), 217
- mqc\_set\_realarray2matrix
  - mqc\_algebra, 140
  - mqc\_algebra::assignment(=), 217
- mqc\_set\_vector2complexarray
  - mqc\_algebra, 141
  - mqc\_algebra::assignment(=), 218
- mqc\_set\_vector2integerarray
  - mqc\_algebra, 142
  - mqc\_algebra::assignment(=), 219
- mqc\_set\_vector2realarray
  - mqc\_algebra, 143
  - mqc\_algebra::assignment(=), 220
- mqc\_set\_vector2vector
  - mqc\_algebra, 143
  - mqc\_algebra::assignment(=), 220
- mqc\_twoeris\_allocate
  - mqc\_est, 196
- mqc\_twoeris\_at
  - mqc\_est, 196
- mqc\_vector2diagmatrix
  - mqc\_algebra, 144
- mqc\_vector\_abs
  - mqc\_algebra, 145
  - mqc\_algebra::abs, 200
- mqc\_vector\_argsort
  - mqc\_algebra, 145
- mqc\_vector\_cast\_complex
  - mqc\_algebra, 146
  - mqc\_algebra::mqc\_cast\_complex, 240
- mqc\_vector\_cast\_integer
  - mqc\_algebra, 147
  - mqc\_algebra::mqc\_cast\_integer, 242
- mqc\_vector\_cast\_real
  - mqc\_algebra, 147
  - mqc\_algebra::mqc\_cast\_real, 243
- mqc\_vector\_cmplx
  - mqc\_algebra, 148
  - mqc\_algebra::cmplx, 225
- mqc\_vector\_complex\_imagpart
  - mqc\_algebra, 149
  - mqc\_algebra::aimag, 203
- mqc\_vector\_complex\_realpart
  - mqc\_algebra, 149
  - mqc\_algebra::real, 355
- mqc\_vector\_conjugate\_transpose
  - mqc\_algebra, 150
  - mqc\_algebra::dagger, 230
- mqc\_vector\_copy\_complex2int
  - mqc\_algebra, 150
- mqc\_vector\_copy\_complex2real
  - mqc\_algebra, 151
- mqc\_vector\_copy\_int2complex
  - mqc\_algebra, 152
- mqc\_vector\_copy\_int2real
  - mqc\_algebra, 152
- mqc\_vector\_copy\_real2complex
  - mqc\_algebra, 153
- mqc\_vector\_copy\_real2int
  - mqc\_algebra, 153
- mqc\_vector\_havecomplex
  - mqc\_algebra, 154
  - mqc\_algebra::mqc\_have\_complex, 247
- mqc\_vector\_haveinteger
  - mqc\_algebra, 155
  - mqc\_algebra::mqc\_have\_int, 248
- mqc\_vector\_havereal
  - mqc\_algebra, 155
  - mqc\_algebra::mqc\_have\_real, 250
- mqc\_vector\_initialize
  - mqc\_algebra, 157
- mqc\_vector\_isallocated
  - mqc\_algebra, 158
- mqc\_vector\_iscolumn
  - mqc\_algebra, 158
- mqc\_vector\_maxloc
  - mqc\_algebra, 159
- mqc\_vector\_maxval
  - mqc\_algebra, 159
- mqc\_vector\_minloc
  - mqc\_algebra, 160
- mqc\_vector\_minval
  - mqc\_algebra, 161
- mqc\_vector\_norm
  - mqc\_algebra, 161



- mqc\_vector\_pop
  - mqc\_algebra, [162](#)
- mqc\_vector\_power
  - mqc\_algebra, [163](#)
- mqc\_vector\_push
  - mqc\_algebra, [164](#)
- mqc\_vector\_scalar\_at
  - mqc\_algebra, [164](#)
- mqc\_vector\_scalar\_increment
  - mqc\_algebra, [165](#)
- mqc\_vector\_scalar\_put
  - mqc\_algebra, [166](#)
- mqc\_vector\_shift
  - mqc\_algebra, [166](#)
- mqc\_vector\_sort
  - mqc\_algebra, [167](#)
- mqc\_vector\_sqrt
  - mqc\_algebra, [168](#)
- mqc\_vector\_transpose
  - mqc\_algebra, [168](#)
  - mqc\_algebra::transpose, [361](#)
- mqc\_vector\_unshift
  - mqc\_algebra, [169](#)
- mqc\_vector\_vector\_at
  - mqc\_algebra, [170](#)
- mqc\_vector\_vector\_put
  - mqc\_algebra, [170](#)
- mqc\_vectorcomplexdivide
  - mqc\_algebra, [171](#)
  - mqc\_algebra::operator(/), [352](#)
- mqc\_vectorcomplexproduct
  - mqc\_algebra, [172](#)
  - mqc\_algebra::operator(\*), [299](#)
- mqc\_vectorintegerdivide
  - mqc\_algebra, [173](#)
  - mqc\_algebra::operator(/), [352](#)
- mqc\_vectorintegerproduct
  - mqc\_algebra, [173](#)
  - mqc\_algebra::operator(\*), [300](#)
- mqc\_vectormatrixdotproduct
  - mqc\_algebra, [174](#)
  - mqc\_algebra::matmul, [234](#)
  - mqc\_algebra::operator(.dot.), [324](#)
- mqc\_vectorrealdive
  - mqc\_algebra, [174](#)
  - mqc\_algebra::operator(/), [353](#)
- mqc\_vectorrealproduct
  - mqc\_algebra, [176](#)
  - mqc\_algebra::operator(\*), [301](#)
- mqc\_vectorscalardivide
  - mqc\_algebra, [177](#)
  - mqc\_algebra::operator(/), [354](#)
- mqc\_vectorscalarproduct
  - mqc\_algebra, [177](#)
  - mqc\_algebra::operator(\*), [301](#)
- mqc\_vectorvectordifference
  - mqc\_algebra, [178](#)
  - mqc\_algebra::operator(-), [322](#)
- mqc\_vectorvectordotproduct
  - mqc\_algebra, [179](#)
  - mqc\_algebra::dot\_product, [232](#)
  - mqc\_algebra::operator(.dot.), [324](#)
- mqc\_vectorvectorsum
  - mqc\_algebra, [180](#)
  - mqc\_algebra::operator(+), [313](#)
- multiplicity
  - mqc\_est::mqc\_wavefunction, [288](#)
- nactive
  - mqc\_est::mqc\_pscf\_wavefunction, [268](#)
- nalpna
  - mqc\_est::mqc\_wavefunction, [288](#)
- nalpstr
  - mqc\_est::mqc\_determinant, [244](#)
- nbasis
  - mqc\_est::mqc\_wavefunction, [288](#)
- nbeta
  - mqc\_est::mqc\_wavefunction, [288](#)
- nbetstr
  - mqc\_est::mqc\_determinant, [244](#)
- ncore
  - mqc\_est::mqc\_pscf\_wavefunction, [269](#)
- ndets
  - mqc\_est::mqc\_determinant, [244](#)
- nelectrons
  - mqc\_est::mqc\_wavefunction, [288](#)
- nfrz
  - mqc\_est::mqc\_pscf\_wavefunction, [269](#)
- norm
  - mqc\_algebra::mqc\_matrix, [254](#)
  - mqc\_algebra::mqc\_vector, [283](#)
  - mqc\_est::mqc\_scf\_integral, [276](#)
- nval
  - mqc\_est::mqc\_pscf\_wavefunction, [269](#)
- orbitals
  - mqc\_est::mqc\_scf\_integral, [276](#)
- order
  - mqc\_est::mqc\_determinant, [245](#)
- overlap\_matrix
  - mqc\_est::mqc\_wavefunction, [288](#)
- pop
  - mqc\_algebra::mqc\_vector, [283](#)
- power
  - mqc\_algebra::mqc\_vector, [283](#)
  - mqc\_est::mqc\_scf\_eigenvalues, [273](#)
- print
  - mqc\_algebra::mqc\_matrix, [254](#)

- mqc\_algebra::mqc\_r4tensor, 270
- mqc\_algebra::mqc\_scalar, 272
- mqc\_algebra::mqc\_vector, 283
- mqc\_est::mqc\_scf\_eigenvalues, 273
- mqc\_est::mqc\_scf\_integral, 276
- mqc\_est::mqc\_twoeris, 280
- mqc\_est::mqc\_wavefunction, 286
- pscf\_amplitudes
  - mqc\_est::mqc\_pscf\_wavefunction, 269
- pscf\_energies
  - mqc\_est::mqc\_pscf\_wavefunction, 269
- push
  - mqc\_algebra::mqc\_vector, 284
- put
  - mqc\_algebra::mqc\_matrix, 254
  - mqc\_algebra::mqc\_r4tensor, 270
  - mqc\_algebra::mqc\_vector, 284
- random
  - mqc\_algebra::mqc\_scalar, 272
- rmsmax
  - mqc\_algebra::mqc\_matrix, 254
- rval
  - mqc\_algebra::mqc\_scalar, 272
- scf\_density\_matrix
  - mqc\_est::mqc\_wavefunction, 289
- set
  - mqc\_algebra::mqc\_matrix, 254
- setelist
  - mqc\_est::mqc\_scf\_integral, 276
- shift
  - mqc\_algebra::mqc\_vector, 284
- size
  - mqc\_algebra::mqc\_vector, 284
- slater\_condon
  - mqc\_est, 196
- sort
  - mqc\_algebra::mqc\_vector, 284
- sqrt
  - mqc\_algebra::mqc\_matrix, 255
  - mqc\_algebra::mqc\_vector, 284
- src/mqc\_algebra.F03, 363
- src/mqc\_est.F03, 376
- strings
  - mqc\_est::mqc\_determinant, 245
- svd
  - mqc\_algebra::mqc\_matrix, 255
- swap
  - mqc\_est::mqc\_scf\_integral, 276
- symindexhash
  - mqc\_algebra, 180
- symmetry
  - mqc\_est::mqc\_wavefunction, 289
- trace
  - mqc\_algebra::mqc\_matrix, 255
  - mqc\_est::mqc\_scf\_integral, 276
- transpose
  - mqc\_algebra::mqc\_matrix, 255
  - mqc\_algebra::mqc\_vector, 285
- twoeri\_trans
  - mqc\_est, 196
- unshift
  - mqc\_algebra::mqc\_vector, 285
- vat
  - mqc\_algebra::mqc\_matrix, 255
  - mqc\_algebra::mqc\_vector, 285
- vput
  - mqc\_algebra::mqc\_matrix, 255
  - mqc\_algebra::mqc\_vector, 285
- wf\_complex
  - mqc\_est::mqc\_wavefunction, 289
- wf\_type
  - mqc\_est::mqc\_wavefunction, 289