

MQCPack

Generated by Doxygen 1.8.16

1 Modules Index	1
1.1 Modules List	1
2 Data Type Index	3
2.1 Class Hierarchy	3
3 Data Type Index	5
3.1 Data Types List	5
4 File Index	7
4.1 File List	7
5 Module Documentation	9
5.1 mqc_algebra Module Reference	9
5.1.1 Function/Subroutine Documentation	16
5.1.1.1 bin_coeff()	16
5.1.1.2 factorial()	17
5.1.1.3 matrix_symm2sq_complex()	17
5.1.1.4 matrix_symm2sq_integer()	18
5.1.1.5 matrix_symm2sq_real()	18
5.1.1.6 mqc_allocate_matrix()	18
5.1.1.7 mqc_allocate_r4tensor()	18
5.1.1.8 mqc_allocate_scalar()	19
5.1.1.9 mqc_allocate_vector()	19
5.1.1.10 mqc_complexscalaradd()	20
5.1.1.11 mqc_complexscalardivide()	20
5.1.1.12 mqc_complexscalarmultiply()	20
5.1.1.13 mqc_complexscalarsubtract()	20
5.1.1.14 mqc_complexvectorproduct()	20
5.1.1.15 mqc_crossproduct()	20
5.1.1.16 mqc_deallocate_matrix()	21
5.1.1.17 mqc_deallocate_r4tensor()	21
5.1.1.18 mqc_deallocate_scalar()	21
5.1.1.19 mqc_deallocate_vector()	21
5.1.1.20 mqc_elementmatrixdivide()	22
5.1.1.21 mqc_elementmatrixproduct()	22
5.1.1.22 mqc_elementvectorproduct()	22
5.1.1.23 mqc_givens_matrix()	22
5.1.1.24 mqc_input_complex_scalar()	22
5.1.1.25 mqc_input_integer_scalar()	23
5.1.1.26 mqc_input_real_scalar()	24

5.1.1.27 <code>mqc_integertscalar()</code>	24
5.1.1.28 <code>mqc_integerlescalar()</code>	25
5.1.1.29 <code>mqc_integerscalaradd()</code>	25
5.1.1.30 <code>mqc_integerscalardivide()</code>	26
5.1.1.31 <code>mqc_integerscalarmultiply()</code>	26
5.1.1.32 <code>mqc_integerscalarsubtract()</code>	26
5.1.1.33 <code>mqc_integervectorproduct()</code>	26
5.1.1.34 <code>mqc_length_vector()</code>	26
5.1.1.35 <code>mqc_matrix_cast_complex()</code>	26
5.1.1.36 <code>mqc_matrix_cast_real()</code>	27
5.1.1.37 <code>mqc_matrix_columns()</code>	27
5.1.1.38 <code>mqc_matrix_conjugate_transpose()</code>	27
5.1.1.39 <code>mqc_matrix_copy_complex2int()</code>	27
5.1.1.40 <code>mqc_matrix_copy_complex2real()</code>	27
5.1.1.41 <code>mqc_matrix_copy_int2complex()</code>	27
5.1.1.42 <code>mqc_matrix_copy_int2real()</code>	28
5.1.1.43 <code>mqc_matrix_copy_real2complex()</code>	28
5.1.1.44 <code>mqc_matrix_copy_real2int()</code>	28
5.1.1.45 <code>mqc_matrix_determinant()</code>	28
5.1.1.46 <code>mqc_matrix_diag2full()</code>	28
5.1.1.47 <code>mqc_matrix_diag2symm()</code>	28
5.1.1.48 <code>mqc_matrix_diagmatrix_put_complex()</code>	29
5.1.1.49 <code>mqc_matrix_diagmatrix_put_integer()</code>	29
5.1.1.50 <code>mqc_matrix_diagmatrix_put_real()</code>	29
5.1.1.51 <code>mqc_matrix_diagmatrix_put_vector()</code>	29
5.1.1.52 <code>mqc_matrix_diagonalize()</code>	29
5.1.1.53 <code>mqc_matrix_full2diag()</code>	29
5.1.1.54 <code>mqc_matrix_full2symm()</code>	30
5.1.1.55 <code>mqc_matrix_generalized_eigensystem()</code>	30
5.1.1.56 <code>mqc_matrix_havecomplex()</code>	30
5.1.1.57 <code>mqc_matrix_havediagonal()</code>	30
5.1.1.58 <code>mqc_matrix_havefull()</code>	30
5.1.1.59 <code>mqc_matrix_haveinteger()</code>	30
5.1.1.60 <code>mqc_matrix_havereal()</code>	31
5.1.1.61 <code>mqc_matrix_havesymmetric()</code>	31
5.1.1.62 <code>mqc_matrix_identity()</code>	31
5.1.1.63 <code>mqc_matrix_initialize()</code>	31
5.1.1.64 <code>mqc_matrix_inverse()</code>	31
5.1.1.65 <code>mqc_matrix_isallocated()</code>	31

5.1.1.66 mqc_matrix_matrix_at()	32
5.1.1.67 mqc_matrix_matrix_contraction()	33
5.1.1.68 mqc_matrix_matrix_put()	33
5.1.1.69 mqc_matrix_norm()	34
5.1.1.70 mqc_matrix_rms_max()	34
5.1.1.71 mqc_matrix_rows()	34
5.1.1.72 mqc_matrix_scalar_at()	34
5.1.1.73 mqc_matrix_scalar_put()	34
5.1.1.74 mqc_matrix_set()	35
5.1.1.75 mqc_matrix_sqrt()	35
5.1.1.76 mqc_matrix_storagetype()	35
5.1.1.77 mqc_matrix_svd()	35
5.1.1.78 mqc_matrix_symm2diag()	35
5.1.1.79 mqc_matrix_symm2full()	36
5.1.1.80 mqc_matrix_symm2full_func()	36
5.1.1.81 mqc_matrix_symmetrize()	36
5.1.1.82 mqc_matrix_symmmatrix_put_complex()	36
5.1.1.83 mqc_matrix_symmmatrix_put_integer()	36
5.1.1.84 mqc_matrix_symmmatrix_put_real()	36
5.1.1.85 mqc_matrix_symmsymmr4tensor_put_complex()	37
5.1.1.86 mqc_matrix_symmsymmr4tensor_put_real()	37
5.1.1.87 mqc_matrix_test_diagonal()	37
5.1.1.88 mqc_matrix_test_symmetric()	37
5.1.1.89 mqc_matrix_trace()	37
5.1.1.90 mqc_matrix_transpose()	37
5.1.1.91 mqc_matrix_vector_at()	38
5.1.1.92 mqc_matrix_vector_put()	38
5.1.1.93 mqc_matrixmatrixdotproduct()	38
5.1.1.94 mqc_matrixmatrixproduct()	38
5.1.1.95 mqc_matrixmatrixsubtract()	38
5.1.1.96 mqc_matrixmatrixsum()	39
5.1.1.97 mqc_matrixscalarproduct()	39
5.1.1.98 mqc_matrixvectordotproduct()	39
5.1.1.99 mqc_outer()	39
5.1.1.100 mqc_output_complex_scalar()	39
5.1.1.101 mqc_output_integer_scalar()	40
5.1.1.102 mqc_output_mqcscalar_scalar()	41
5.1.1.103 mqc_output_real_scalar()	41
5.1.1.104 mqc_print_matrix_algebra1()	42

5.1.1.105 mqc_print_r4tensor_algebra1()	42
5.1.1.106 mqc_print_scalar_algebra1()	43
5.1.1.107 mqc_print_vector_algebra1()	44
5.1.1.108 mqc_r4tensor_at()	44
5.1.1.109 mqc_r4tensor_havecomplex()	44
5.1.1.110 mqc_r4tensor_haveinteger()	44
5.1.1.111 mqc_r4tensor_havereal()	44
5.1.1.112 mqc_r4tensor_initialize()	45
5.1.1.113 mqc_r4tensor_put()	45
5.1.1.114 mqc_realgtscalar()	45
5.1.1.115 mqc_realltscalar()	45
5.1.1.116 mqc_realltscalar()	46
5.1.1.117 mqc_realscalaradd()	46
5.1.1.118 mqc_realscalddivide()	47
5.1.1.119 mqc_realscalarmultiply()	47
5.1.1.120 mqc_realscalarsubtract()	47
5.1.1.121 mqc_realvectorproduct()	47
5.1.1.122 mqc_scalar_acos()	47
5.1.1.123 mqc_scalar_asin()	48
5.1.1.124 mqc_scalar_atan()	49
5.1.1.125 mqc_scalar_atan2()	49
5.1.1.126 mqc_scalar_cmplx()	50
5.1.1.127 mqc_scalar_complex_conjugate()	50
5.1.1.128 mqc_scalar_complex_imagpart()	51
5.1.1.129 mqc_scalar_complex_realpart()	51
5.1.1.130 mqc_scalar_cos()	51
5.1.1.131 mqc_scalar_get_abs_value()	51
5.1.1.132 mqc_scalar_get_intrinsic_complex()	52
5.1.1.133 mqc_scalar_get_intrinsic_integer()	53
5.1.1.134 mqc_scalar_get_intrinsic_real()	53
5.1.1.135 mqc_scalar_get_random_value()	54
5.1.1.136 mqc_scalar_havecomplex()	55
5.1.1.137 mqc_scalar_haveinteger()	55
5.1.1.138 mqc_scalar_havereal()	56
5.1.1.139 mqc_scalar_isallocated()	57
5.1.1.140 mqc_scalar_sin()	57
5.1.1.141 mqc_scalar_sqrt()	58
5.1.1.142 mqc_scalar_tan()	58
5.1.1.143 mqc_scalaradd()	59

5.1.1.144 mqc_scalarcomplexadd()	60
5.1.1.145 mqc_scalarcomplexdivide()	60
5.1.1.146 mqc_scalarcomplexexponent()	60
5.1.1.147 mqc_scalarcomplexmultiply()	61
5.1.1.148 mqc_scalarcomplexsubtract()	61
5.1.1.149 mqc_scalardivide()	61
5.1.1.150 mqc_scalareq()	62
5.1.1.151 mqc_scalarexponent()	62
5.1.1.152 mqc_scalarge()	63
5.1.1.153 mqc_scalargt()	63
5.1.1.154 mqc_scalargtinteger()	64
5.1.1.155 mqc_scalargtreal()	65
5.1.1.156 mqc_scalarintegeradd()	65
5.1.1.157 mqc_scalarintegerdivide()	65
5.1.1.158 mqc_scalarintegerexponent()	65
5.1.1.159 mqc_scalarintegermultiply()	66
5.1.1.160 mqc_scalarintegersubtract()	66
5.1.1.161 mqc_scalarle()	66
5.1.1.162 mqc_scalarleinteger()	67
5.1.1.163 mqc_scalarlereal()	67
5.1.1.164 mqc_scalarlt()	67
5.1.1.165 mqc_scalarltreal()	68
5.1.1.166 mqc_scalarmatrixproduct()	68
5.1.1.167 mqc_scalarmultiply()	69
5.1.1.168 mqc_scalarne()	69
5.1.1.169 mqc_scalarrealadd()	70
5.1.1.170 mqc_scalarrealddivide()	70
5.1.1.171 mqc_scalarrealexponent()	70
5.1.1.172 mqc_scalarrealmultiply()	71
5.1.1.173 mqc_scalarrealsubtract()	71
5.1.1.174 mqc_scalarsubtract()	71
5.1.1.175 mqc_scalarvectordifference()	72
5.1.1.176 mqc_scalarvectorproduct()	72
5.1.1.177 mqc_scalarvectorsum()	72
5.1.1.178 mqc_set_array2tensor()	73
5.1.1.179 mqc_set_array2vector_complex()	73
5.1.1.180 mqc_set_array2vector_integer()	73
5.1.1.181 mqc_set_array2vector_real()	73
5.1.1.182 mqc_set_complexarray2matrix()	73

5.1.1.183 mqc_set_integerarray2matrix()	73
5.1.1.184 mqc_set_matrix2complexarray()	74
5.1.1.185 mqc_set_matrix2integerarray()	74
5.1.1.186 mqc_set_matrix2matrix()	74
5.1.1.187 mqc_set_matrix2realarray()	74
5.1.1.188 mqc_set_realarray2matrix()	74
5.1.1.189 mqc_set_vector2complexarray()	74
5.1.1.190 mqc_set_vector2integerarray()	75
5.1.1.191 mqc_set_vector2realarray()	75
5.1.1.192 mqc_set_vector2vector()	75
5.1.1.193 mqc_vector2diagmatrix()	75
5.1.1.194 mqc_vector_abs()	75
5.1.1.195 mqc_vector_argsort()	75
5.1.1.196 mqc_vector_cast_complex()	76
5.1.1.197 mqc_vector_cast_real()	76
5.1.1.198 mqc_vector_cmplx()	76
5.1.1.199 mqc_vector_complex_imagpart()	76
5.1.1.200 mqc_vector_complex_realpart()	76
5.1.1.201 mqc_vector_conjugate_transpose()	76
5.1.1.202 mqc_vector_copy_complex2int()	77
5.1.1.203 mqc_vector_copy_complex2real()	77
5.1.1.204 mqc_vector_copy_int2complex()	77
5.1.1.205 mqc_vector_copy_int2real()	77
5.1.1.206 mqc_vector_copy_real2complex()	77
5.1.1.207 mqc_vector_copy_real2int()	77
5.1.1.208 mqc_vector_havecomplex()	78
5.1.1.209 mqc_vector_haveinteger()	78
5.1.1.210 mqc_vector_havereal()	78
5.1.1.211 mqc_vector_initialize()	78
5.1.1.212 mqc_vector_isallocated()	78
5.1.1.213 mqc_vector_iscolumn()	78
5.1.1.214 mqc_vector_maxloc()	79
5.1.1.215 mqc_vector_maxval()	79
5.1.1.216 mqc_vector_minloc()	79
5.1.1.217 mqc_vector_minval()	79
5.1.1.218 mqc_vector_norm()	79
5.1.1.219 mqc_vector_pop()	79
5.1.1.220 mqc_vector_power()	80
5.1.1.221 mqc_vector_push()	80

5.1.1.222 mqc_vector_scalar_at()	80
5.1.1.223 mqc_vector_scalar_increment()	80
5.1.1.224 mqc_vector_scalar_put()	80
5.1.1.225 mqc_vector_shift()	80
5.1.1.226 mqc_vector_sort()	81
5.1.1.227 mqc_vector_sqrt()	81
5.1.1.228 mqc_vector_transpose()	81
5.1.1.229 mqc_vector_unshift()	81
5.1.1.230 mqc_vector_vector_at()	81
5.1.1.231 mqc_vector_vector_put()	81
5.1.1.232 mqc_vectorcomplexdivide()	82
5.1.1.233 mqc_vectorcomplexproduct()	82
5.1.1.234 mqc_vectorintegerdivide()	82
5.1.1.235 mqc_vectorintegerproduct()	82
5.1.1.236 mqc_vectormatrixdotproduct()	82
5.1.1.237 mqc_vectorrealdivide()	82
5.1.1.238 mqc_vectorrealproduct()	83
5.1.1.239 mqc_vectorscalardivide()	83
5.1.1.240 mqc_vectorscalarproduct()	83
5.1.1.241 mqc_vectorvectordifference()	83
5.1.1.242 mqc_vectorvectordotproduct()	83
5.1.1.243 mqc_vectorvectorsum()	83
5.1.1.244 symindexhash()	84
5.2 mqc_est Module Reference	84
5.2.1 Function/Subroutine Documentation	86
5.2.1.1 gen_det_str()	86
5.2.1.2 get_one_gamma_matrix()	86
5.2.1.3 mqc_build_ci_hamiltonian()	86
5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()	87
5.2.1.5 mqc_eigenvalues_add_name()	87
5.2.1.6 mqc_eigenvalues_allocate()	87
5.2.1.7 mqc_eigenvalues_array_name()	87
5.2.1.8 mqc_eigenvalues_array_type()	87
5.2.1.9 mqc_eigenvalues_at()	87
5.2.1.10 mqc_eigenvalues_dimension()	88
5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()	88
5.2.1.12 mqc_eigenvalues_has_alpha()	88
5.2.1.13 mqc_eigenvalues_has_beta()	88
5.2.1.14 mqc_eigenvalues_integral_multiply()	88

5.2.1.15 mqc_eigenvalues_isallocated()	88
5.2.1.16 mqc_eigenvalues_output_array()	89
5.2.1.17 mqc_eigenvalues_output_block()	89
5.2.1.18 mqc_eri_integral_contraction()	89
5.2.1.19 mqc_integral_add_name()	89
5.2.1.20 mqc_integral_allocate()	89
5.2.1.21 mqc_integral_array_name()	90
5.2.1.22 mqc_integral_array_type()	90
5.2.1.23 mqc_integral_at()	90
5.2.1.24 mqc_integral_conjugate_transpose()	90
5.2.1.25 mqc_integral_delete_energy_list()	90
5.2.1.26 mqc_integral_difference()	90
5.2.1.27 mqc_integral_dimension()	91
5.2.1.28 mqc_integral_eigenvalues_multiply()	91
5.2.1.29 mqc_integral_get_energy_list()	91
5.2.1.30 mqc_integral_has_alpha()	91
5.2.1.31 mqc_integral_has_alphabeta()	91
5.2.1.32 mqc_integral_has_beta()	91
5.2.1.33 mqc_integral_has_betaalpha()	92
5.2.1.34 mqc_integral_identity()	92
5.2.1.35 mqc_integral_initialize()	92
5.2.1.36 mqc_integral_integral_multiply()	92
5.2.1.37 mqc_integral_isallocated()	92
5.2.1.38 mqc_integral_matrix_multiply()	93
5.2.1.39 mqc_integral_norm()	93
5.2.1.40 mqc_integral_output_array()	93
5.2.1.41 mqc_integral_output_block()	93
5.2.1.42 mqc_integral_output_orbitals()	93
5.2.1.43 mqc_integral_scalar_multiply()	94
5.2.1.44 mqc_integral_set_energy_list()	94
5.2.1.45 mqc_integral_sum()	94
5.2.1.46 mqc_integral_swap_orbitals()	94
5.2.1.47 mqc_integral_transpose()	94
5.2.1.48 mqc_matrix_integral_multiply()	95
5.2.1.49 mqc_matrix_spinblockghf()	95
5.2.1.50 mqc_matrix_undospinblockghf_eigenvalues()	95
5.2.1.51 mqc_matrix_undospinblockghf_integral()	95
5.2.1.52 mqc_print_eigenvalues()	95
5.2.1.53 mqc_print_integral()	96

5.2.1.54 mqc_print_twoeris()	96
5.2.1.55 mqc_print_wavefunction()	96
5.2.1.56 mqc_scalar_integral_multiply()	96
5.2.1.57 mqc_scf_eigenvalues_power()	96
5.2.1.58 mqc_scf_integral_contraction()	97
5.2.1.59 mqc_scf_integral_determinant()	97
5.2.1.60 mqc_scf_integral_diagonalize()	97
5.2.1.61 mqc_scf_integral_generalized_eigensystem()	97
5.2.1.62 mqc_scf_integral_inverse()	97
5.2.1.63 mqc_scf_integral_trace()	97
5.2.1.64 mqc_scf_transformation_matrix()	98
5.2.1.65 mqc_twoeris_allocate()	98
5.2.1.66 mqc_twoeris_at()	98
5.2.1.67 slater_condon()	98
5.2.1.68 twoeri_trans()	99
6 Data Type Documentation	101
6.1 mqc_algebra::abs Interface Reference	101
6.1.1 Member Function/Subroutine Documentation	101
6.1.1.1 mqc_scalar_get_abs_value()	101
6.1.1.2 mqc_vector_abs()	102
6.2 mqc_algebra::acos Interface Reference	102
6.2.1 Member Function/Subroutine Documentation	102
6.2.1.1 mqc_scalar_acos()	102
6.3 mqc_algebra::aimag Interface Reference	103
6.3.1 Member Function/Subroutine Documentation	103
6.3.1.1 mqc_scalar_complex_imagpart()	103
6.3.1.2 mqc_vector_complex_imagpart()	103
6.4 mqc_algebra::asin Interface Reference	104
6.4.1 Member Function/Subroutine Documentation	104
6.4.1.1 mqc_scalar_asin()	104
6.5 mqc_algebra::assignment(=) Interface Reference	105
6.5.1 Member Function/Subroutine Documentation	105
6.5.1.1 mqc_input_complex_scalar()	105
6.5.1.2 mqc_input_integer_scalar()	106
6.5.1.3 mqc_input_real_scalar()	107
6.5.1.4 mqc_output_complex_scalar()	107
6.5.1.5 mqc_output_integer_scalar()	108
6.5.1.6 mqc_output_mqcscalar_scalar()	109

6.5.1.7 mqc_output_real_scalar()	109
6.5.1.8 mqc_set_array2tensor()	110
6.5.1.9 mqc_set_array2vector_complex()	110
6.5.1.10 mqc_set_array2vector_integer()	110
6.5.1.11 mqc_set_array2vector_real()	111
6.5.1.12 mqc_set_complexarray2matrix()	111
6.5.1.13 mqc_set_integerarray2matrix()	111
6.5.1.14 mqc_set_matrix2complexarray()	111
6.5.1.15 mqc_set_matrix2integerarray()	111
6.5.1.16 mqc_set_matrix2matrix()	111
6.5.1.17 mqc_set_matrix2realarray()	112
6.5.1.18 mqc_set_realarray2matrix()	112
6.5.1.19 mqc_set_vector2complexarray()	112
6.5.1.20 mqc_set_vector2integerarray()	112
6.5.1.21 mqc_set_vector2realarray()	112
6.5.1.22 mqc_set_vector2vector()	112
6.6 mqc_est::assignment(=) Interface Reference	113
6.6.1 Member Function/Subroutine Documentation	113
6.6.1.1 mqc_eigenvalues_output_array()	113
6.6.1.2 mqc_integral_output_array()	113
6.7 mqc_algebra::atan Interface Reference	113
6.7.1 Member Function/Subroutine Documentation	113
6.7.1.1 mqc_scalar_atan()	113
6.8 mqc_algebra::atan2 Interface Reference	114
6.8.1 Member Function/Subroutine Documentation	114
6.8.1.1 mqc_scalar_atan2()	114
6.9 mqc_algebra::cmplx Interface Reference	115
6.9.1 Member Function/Subroutine Documentation	115
6.9.1.1 mqc_scalar_cmplx()	115
6.9.1.2 mqc_vector_cmplx()	116
6.10 mqc_algebra::conjg Interface Reference	116
6.10.1 Member Function/Subroutine Documentation	116
6.10.1.1 mqc_scalar_complex_conjugate()	117
6.11 mqc_algebra::contraction Interface Reference	117
6.11.1 Member Function/Subroutine Documentation	117
6.11.1.1 mqc_matrix_matrix_contraction()	117
6.12 mqc_est::contraction Interface Reference	117
6.12.1 Member Function/Subroutine Documentation	117
6.12.1.1 mqc_eri_integral_contraction()	118

6.12.1.2 mqc_scf_integral_contraction()	118
6.13 mqc_algebra::cos Interface Reference	118
6.13.1 Member Function/Subroutine Documentation	118
6.13.1.1 mqc_scalar_cos()	118
6.14 mqc_algebra::dagger Interface Reference	119
6.14.1 Member Function/Subroutine Documentation	119
6.14.1.1 mqc_matrix_conjugate_transpose()	119
6.14.1.2 mqc_vector_conjugate_transpose()	119
6.15 mqc_est::dagger Interface Reference	120
6.15.1 Member Function/Subroutine Documentation	120
6.15.1.1 mqc_integral_conjugate_transpose()	120
6.16 mqc_algebra::dot_product Interface Reference	120
6.16.1 Member Function/Subroutine Documentation	120
6.16.1.1 mqc_vectorvectordotproduct()	120
6.17 mqc_est::dot_product Interface Reference	121
6.17.1 Member Function/Subroutine Documentation	121
6.17.1.1 mqc_eigenvalue_eigenvalue_dotproduct()	121
6.18 mqc_algebra::matmul Interface Reference	121
6.18.1 Member Function/Subroutine Documentation	121
6.18.1.1 mqc_matrixmatrixdotproduct()	121
6.18.1.2 mqc_matrixvectordotproduct()	122
6.18.1.3 mqc_vectormatrixdotproduct()	122
6.19 mqc_est::matmul Interface Reference	122
6.19.1 Member Function/Subroutine Documentation	122
6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()	122
6.19.1.2 mqc_eigenvalues_integral_multiply()	123
6.19.1.3 mqc_integral_eigenvalues_multiply()	123
6.19.1.4 mqc_integral_integral_multiply()	123
6.19.1.5 mqc_integral_matrix_multiply()	123
6.19.1.6 mqc_matrix_integral_multiply()	123
6.20 mqc_algebra::matrix_symm2sq Interface Reference	124
6.20.1 Member Function/Subroutine Documentation	124
6.20.1.1 matrix_symm2sq_complex()	124
6.20.1.2 matrix_symm2sq_integer()	124
6.20.1.3 matrix_symm2sq_real()	124
6.21 mqc_algebra::mqc_cast_complex Interface Reference	125
6.21.1 Member Function/Subroutine Documentation	125
6.21.1.1 mqc_matrix_cast_complex()	125
6.21.1.2 mqc_vector_cast_complex()	125

6.22 mqc_algebra::mqc_cast_real Interface Reference	125
6.22.1 Member Function/Subroutine Documentation	125
6.22.1.1 mqc_matrix_cast_real()	126
6.22.1.2 mqc_vector_cast_real()	126
6.23 mqc_est::mqc_determinant Type Reference	126
6.23.1 Member Data Documentation	126
6.23.1.1 nalpstr	126
6.23.1.2 nbetstr	126
6.23.1.3 ndets	127
6.23.1.4 order	127
6.23.1.5 strings	127
6.24 mqc_est::mqc_determinant_string Type Reference	127
6.24.1 Member Data Documentation	127
6.24.1.1 alpha	127
6.24.1.2 beta	127
6.25 mqc_algebra::mqc_have_complex Interface Reference	128
6.25.1 Member Function/Subroutine Documentation	128
6.25.1.1 mqc_matrix_havecomplex()	128
6.25.1.2 mqc_vector_havecomplex()	128
6.26 mqc_algebra::mqc_have_int Interface Reference	128
6.26.1 Member Function/Subroutine Documentation	128
6.26.1.1 mqc_matrix_haveinteger()	129
6.26.1.2 mqc_vector_haveinteger()	129
6.27 mqc_algebra::mqc_have_real Interface Reference	129
6.27.1 Member Function/Subroutine Documentation	129
6.27.1.1 mqc_matrix_havereal()	129
6.27.1.2 mqc_vector_havereal()	129
6.28 mqc_algebra::mqc_matrix Type Reference	130
6.28.1 Member Function/Subroutine Documentation	130
6.28.1.1 at()	130
6.28.1.2 dagger()	131
6.28.1.3 det()	131
6.28.1.4 diag()	131
6.28.1.5 eigensys()	131
6.28.1.6 identity()	131
6.28.1.7 init()	131
6.28.1.8 initialize()	131
6.28.1.9 inv()	132
6.28.1.10 mat()	132

6.28.1.11 mput()	132
6.28.1.12 norm()	132
6.28.1.13 print()	132
6.28.1.14 put()	132
6.28.1.15 rmsmax()	132
6.28.1.16 s_type()	133
6.28.1.17 set()	133
6.28.1.18 sqrt()	133
6.28.1.19 svd()	133
6.28.1.20 trace()	133
6.28.1.21 transpose()	133
6.28.1.22 vat()	133
6.28.1.23 vput()	134
6.28.2 Member Data Documentation	134
6.28.2.1 matc	134
6.28.2.2 mati	134
6.28.2.3 matr	134
6.29 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference	134
6.29.1 Member Function/Subroutine Documentation	134
6.29.1.1 mqc_matrix_diagmatrix_put_complex()	135
6.29.1.2 mqc_matrix_diagmatrix_put_integer()	135
6.29.1.3 mqc_matrix_diagmatrix_put_real()	135
6.29.1.4 mqc_matrix_diagmatrix_put_vector()	135
6.30 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference	135
6.30.1 Member Function/Subroutine Documentation	136
6.30.1.1 mqc_matrix_symmmatrix_put_complex()	136
6.30.1.2 mqc_matrix_symmmatrix_put_integer()	136
6.30.1.3 mqc_matrix_symmmatrix_put_real()	136
6.31 mqc_est::mqc_matrix_undospinblockghf Interface Reference	136
6.31.1 Member Function/Subroutine Documentation	136
6.31.1.1 mqc_matrix_undospinblockghf_eigenvalues()	137
6.31.1.2 mqc_matrix_undospinblockghf_integral()	137
6.32 mqc_algebra::mqc_print Interface Reference	137
6.32.1 Member Function/Subroutine Documentation	137
6.32.1.1 mqc_print_matrix_algebra1()	137
6.32.1.2 mqc_print_r4tensor_algebra1()	138
6.32.1.3 mqc_print_scalar_algebra1()	138
6.32.1.4 mqc_print_vector_algebra1()	139
6.33 mqc_est::mqc_print Interface Reference	139

6.33.1 Member Function/Subroutine Documentation	139
6.33.1.1 <code>mqc_print_eigenvalues()</code>	139
6.33.1.2 <code>mqc_print_integral()</code>	140
6.33.1.3 <code>mqc_print_twoeris()</code>	140
6.33.1.4 <code>mqc_print_wavefunction()</code>	140
6.34 <code>mqc_est::mqc_pscf_wavefunction</code> Type Reference	140
6.34.1 Member Data Documentation	141
6.34.1.1 <code>nactive</code>	141
6.34.1.2 <code>ncore</code>	141
6.34.1.3 <code>nfrz</code>	141
6.34.1.4 <code>nval</code>	141
6.34.1.5 <code>pscf_amplitudes</code>	141
6.34.1.6 <code>pscf_energies</code>	142
6.35 <code>mqc_algebra::mqc_r4tensor</code> Type Reference	142
6.35.1 Member Function/Subroutine Documentation	142
6.35.1.1 <code>at()</code>	142
6.35.1.2 <code>init()</code>	142
6.35.1.3 <code>initialize()</code>	142
6.35.1.4 <code>print()</code>	143
6.35.1.5 <code>put()</code>	143
6.36 <code>mqc_algebra::mqc_scalar</code> Type Reference	143
6.36.1 Member Function/Subroutine Documentation	143
6.36.1.1 <code>abs()</code>	143
6.36.1.2 <code>cval()</code>	143
6.36.1.3 <code>ival()</code>	144
6.36.1.4 <code>print()</code>	144
6.36.1.5 <code>random()</code>	144
6.36.1.6 <code>rval()</code>	144
6.37 <code>mqc_est::mqc_scf_eigenvalues</code> Type Reference	144
6.37.1 Member Function/Subroutine Documentation	144
6.37.1.1 <code>addlabel()</code>	145
6.37.1.2 <code>at()</code>	145
6.37.1.3 <code>getblock()</code>	145
6.37.1.4 <code>getlabel()</code>	145
6.37.1.5 <code>power()</code>	145
6.37.1.6 <code>print()</code>	145
6.38 <code>mqc_est::mqc_scf_integral</code> Type Reference	146
6.38.1 Member Function/Subroutine Documentation	146
6.38.1.1 <code>addlabel()</code>	146

6.38.1.2 deleteelist()	146
6.38.1.3 det()	146
6.38.1.4 diag()	147
6.38.1.5 eigensys()	147
6.38.1.6 getblock()	147
6.38.1.7 getelist()	147
6.38.1.8 getlabel()	147
6.38.1.9 identity()	147
6.38.1.10 init()	147
6.38.1.11 inv()	148
6.38.1.12 norm()	148
6.38.1.13 orbitals()	148
6.38.1.14 print()	148
6.38.1.15 setelist()	148
6.38.1.16 swap()	148
6.38.1.17 trace()	148
6.39 mqc_algebra::mqc_set_array2vector Interface Reference	149
6.39.1 Member Function/Subroutine Documentation	149
6.39.1.1 mqc_set_array2vector_complex()	149
6.39.1.2 mqc_set_array2vector_integer()	149
6.39.1.3 mqc_set_array2vector_real()	149
6.40 mqc_est::mqc_twoeris Type Reference	149
6.40.1 Member Function/Subroutine Documentation	150
6.40.1.1 print()	150
6.41 mqc_algebra::mqc_vector Type Reference	150
6.41.1 Member Function/Subroutine Documentation	151
6.41.1.1 abs()	151
6.41.1.2 argsort()	151
6.41.1.3 at()	151
6.41.1.4 dagger()	151
6.41.1.5 diag()	151
6.41.1.6 init()	152
6.41.1.7 initialize()	152
6.41.1.8 maxloc()	152
6.41.1.9 maxval()	152
6.41.1.10 minloc()	152
6.41.1.11 minval()	152
6.41.1.12 norm()	152
6.41.1.13 pop()	153

6.41.1.14 power()	153
6.41.1.15 print()	153
6.41.1.16 push()	153
6.41.1.17 put()	153
6.41.1.18 shift()	153
6.41.1.19 size()	153
6.41.1.20 sort()	154
6.41.1.21 sqrt()	154
6.41.1.22 transpose()	154
6.41.1.23 unshift()	154
6.41.1.24 vat()	154
6.41.1.25 vput()	154
6.41.2 Member Data Documentation	154
6.41.2.1 data_type	154
6.41.2.2 length	155
6.41.2.3 vecc	155
6.41.2.4 veci	155
6.41.2.5 vecr	155
6.42 mqc_est::mqc_wavefunction Type Reference	155
6.42.1 Member Function/Subroutine Documentation	156
6.42.1.1 print()	156
6.42.2 Member Data Documentation	156
6.42.2.1 basis	156
6.42.2.2 charge	156
6.42.2.3 core_hamiltonian	157
6.42.2.4 density_matrix	157
6.42.2.5 fock_matrix	157
6.42.2.6 mo_coefficients	157
6.42.2.7 mo_energies	157
6.42.2.8 mo_symmetries	157
6.42.2.9 multiplicity	157
6.42.2.10 nalpha	158
6.42.2.11 nbasis	158
6.42.2.12 nbeta	158
6.42.2.13 nelectrons	158
6.42.2.14 overlap_matrix	158
6.42.2.15 scf_density_matrix	158
6.42.2.16 symmetry	158
6.42.2.17 wf_complex	159

6.42.2.18 wf_type	159
6.43 mqc_algebra::operator(*) Interface Reference	159
6.43.1 Member Function/Subroutine Documentation	159
6.43.1.1 mqc_complexscalarmultiply()	160
6.43.1.2 mqc_complexvectorproduct()	160
6.43.1.3 mqc_integerscalarmultiply()	160
6.43.1.4 mqc_integervectorproduct()	160
6.43.1.5 mqc_matrixmatrixproduct()	160
6.43.1.6 mqc_matrixscalarproduct()	160
6.43.1.7 mqc_realscalarmultiply()	161
6.43.1.8 mqc_realvectorproduct()	161
6.43.1.9 mqc_scalarcomplexmultiply()	161
6.43.1.10 mqc_scalarintegermultiply()	161
6.43.1.11 mqc_scalarmatrixproduct()	161
6.43.1.12 mqc_scalarmultiply()	161
6.43.1.13 mqc_scalarrealmultiply()	162
6.43.1.14 mqc_scalarvectorproduct()	162
6.43.1.15 mqc_vectorcomplexproduct()	162
6.43.1.16 mqc_vectorintegerproduct()	163
6.43.1.17 mqc_vectorrealproduct()	163
6.43.1.18 mqc_vectorscalarproduct()	163
6.44 mqc_est::operator(*) Interface Reference	163
6.44.1 Member Function/Subroutine Documentation	163
6.44.1.1 mqc_integral_scalar_multiply()	163
6.44.1.2 mqc_scalar_integral_multiply()	164
6.45 mqc_algebra::operator(**) Interface Reference	164
6.45.1 Member Function/Subroutine Documentation	164
6.45.1.1 mqc_scalarcomplexexponent()	164
6.45.1.2 mqc_scalarexponent()	165
6.45.1.3 mqc_scalarintegerexponent()	166
6.45.1.4 mqc_scalarrealexponent()	166
6.46 mqc_est::operator(+) Interface Reference	167
6.46.1 Member Function/Subroutine Documentation	167
6.46.1.1 mqc_integral_sum()	167
6.47 mqc_algebra::operator(+) Interface Reference	168
6.47.1 Member Function/Subroutine Documentation	168
6.47.1.1 mqc_complexscalaradd()	168
6.47.1.2 mqc_integerscalaradd()	168
6.47.1.3 mqc_matrixmatrixsum()	168

6.47.1.4 mqc_realscalaradd()	169
6.47.1.5 mqc_scalaradd()	169
6.47.1.6 mqc_scalarcomplexadd()	169
6.47.1.7 mqc_scalarintegeradd()	170
6.47.1.8 mqc_scalarrealadd()	170
6.47.1.9 mqc_scalarvectorsum()	170
6.47.1.10 mqc_vectorvectorsum()	170
6.48 mqc_est::operator(-) Interface Reference	170
6.48.1 Member Function/Subroutine Documentation	170
6.48.1.1 mqc_integral_difference()	171
6.49 mqc_algebra::operator(-) Interface Reference	171
6.49.1 Member Function/Subroutine Documentation	171
6.49.1.1 mqc_complexscalarsubtract()	171
6.49.1.2 mqc_integerscalarsubtract()	171
6.49.1.3 mqc_matrixmatrixsubtract()	172
6.49.1.4 mqc_realscalarsubtract()	172
6.49.1.5 mqc_scalarcomplexsubtract()	172
6.49.1.6 mqc_scalarintegersubtract()	172
6.49.1.7 mqc_scalarrealsubtract()	172
6.49.1.8 mqc_scalarsubtract()	172
6.49.1.9 mqc_scalarvectordifference()	173
6.49.1.10 mqc_vectorvectordifference()	173
6.50 mqc_algebra::operator(.dot.) Interface Reference	173
6.50.1 Member Function/Subroutine Documentation	174
6.50.1.1 mqc_matrixmatrixdotproduct()	174
6.50.1.2 mqc_matrixvectordotproduct()	174
6.50.1.3 mqc_vectormatrixdotproduct()	174
6.50.1.4 mqc_vectorvectordotproduct()	174
6.51 mqc_algebra::operator(.eq.) Interface Reference	174
6.51.1 Member Function/Subroutine Documentation	175
6.51.1.1 mqc_scalareq()	175
6.52 mqc_algebra::operator(.ewd.) Interface Reference	175
6.52.1 Member Function/Subroutine Documentation	176
6.52.1.1 mqc_elementmatrixdivide()	176
6.53 mqc_algebra::operator(.ewp.) Interface Reference	176
6.53.1 Member Function/Subroutine Documentation	176
6.53.1.1 mqc_elementmatrixproduct()	176
6.53.1.2 mqc_elementvectorproduct()	176
6.54 mqc_algebra::operator(.ge.) Interface Reference	177

6.54.1 Member Function/Subroutine Documentation	177
6.54.1.1 mqc_scalarge()	177
6.55 mqc_algebra::operator(.gt.) Interface Reference	177
6.55.1 Member Function/Subroutine Documentation	177
6.55.1.1 mqc_integergtscalar()	178
6.55.1.2 mqc_realgtscalar()	178
6.55.1.3 mqc_scalargt()	179
6.55.1.4 mqc_scalargtinteger()	179
6.55.1.5 mqc_scalargtreal()	180
6.56 mqc_algebra::operator(.le.) Interface Reference	180
6.56.1 Member Function/Subroutine Documentation	181
6.56.1.1 mqc_integerlescalar()	181
6.56.1.2 mqc_reallescalar()	181
6.56.1.3 mqc_scalarle()	181
6.56.1.4 mqc_scalarleinteger()	181
6.56.1.5 mqc_scalarlereal()	181
6.57 mqc_algebra::operator(.lt.) Interface Reference	182
6.57.1 Member Function/Subroutine Documentation	182
6.57.1.1 mqc_realltscalar()	182
6.57.1.2 mqc_scalarlt()	183
6.57.1.3 mqc_scalarltreal()	184
6.58 mqc_algebra::operator(.ne.) Interface Reference	184
6.58.1 Member Function/Subroutine Documentation	185
6.58.1.1 mqc_scalarne()	185
6.59 mqc_algebra::operator(.outer.) Interface Reference	185
6.59.1 Member Function/Subroutine Documentation	186
6.59.1.1 mqc_outer()	186
6.60 mqc_algebra::operator(.x.) Interface Reference	186
6.60.1 Member Function/Subroutine Documentation	186
6.60.1.1 mqc_crossproduct()	186
6.61 mqc_algebra::operator(/) Interface Reference	186
6.61.1 Member Function/Subroutine Documentation	187
6.61.1.1 mqc_complexscalardivide()	187
6.61.1.2 mqc_integerscalardivide()	187
6.61.1.3 mqc_realscalardivide()	187
6.61.1.4 mqc_scalarcomplexdivide()	187
6.61.1.5 mqc_scalardivide()	187
6.61.1.6 mqc_scalarintegerdivide()	188
6.61.1.7 mqc_scalarrealdivide()	188

6.61.1.8 mqc_vectorcomplexdivide()	188
6.61.1.9 mqc_vectorintegerdivide()	189
6.61.1.10 mqc_vectorrealddivide()	189
6.61.1.11 mqc_vectorscalardivide()	189
6.62 mqc_algebra::real Interface Reference	189
6.62.1 Member Function/Subroutine Documentation	189
6.62.1.1 mqc_scalar_complex_realpart()	189
6.62.1.2 mqc_vector_complex_realpart()	190
6.63 mqc_algebra::sin Interface Reference	190
6.63.1 Member Function/Subroutine Documentation	190
6.63.1.1 mqc_scalar_sin()	190
6.64 mqc_algebra::sqrt Interface Reference	191
6.64.1 Member Function/Subroutine Documentation	191
6.64.1.1 mqc_scalar_sqrt()	191
6.65 mqc_algebra::tan Interface Reference	192
6.65.1 Member Function/Subroutine Documentation	192
6.65.1.1 mqc_scalar_tan()	192
6.66 mqc_est::transpose Interface Reference	193
6.66.1 Member Function/Subroutine Documentation	193
6.66.1.1 mqc_integral_transpose()	193
6.67 mqc_algebra::transpose Interface Reference	193
6.67.1 Member Function/Subroutine Documentation	193
6.67.1.1 mqc_matrix_transpose()	194
6.67.1.2 mqc_vector_transpose()	194
7 File Documentation	195
7.1 src/mqc_algebra.F03 File Reference	195
7.2 src/mqc_est.F03 File Reference	202
Index	205

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

mqc_algebra	9
mqc_est	84

Chapter 2

Data Type Index

2.1 Class Hierarchy

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

<code>mqc_algebra::abs</code>	101
<code>mqc_algebra::acos</code>	102
<code>mqc_algebra::aimag</code>	103
<code>mqc_algebra::asin</code>	104
<code>mqc_algebra::assignment(=)</code>	105
<code>mqc_est::assignment(=)</code>	113
<code>mqc_algebra::atan</code>	113
<code>mqc_algebra::atan2</code>	114
<code>mqc_algebra::cmplx</code>	115
<code>mqc_algebra::conjg</code>	116
<code>mqc_algebra::contraction</code>	117
<code>mqc_est::contraction</code>	117
<code>mqc_algebra::cos</code>	118
<code>mqc_algebra::dagger</code>	119
<code>mqc_est::dagger</code>	120
<code>mqc_algebra::dot_product</code>	120
<code>mqc_est::dot_product</code>	121
<code>mqc_algebra::matmul</code>	121
<code>mqc_est::matmul</code>	122
<code>mqc_algebra::matrix_symm2sq</code>	124
<code>mqc_algebra::mqc_cast_complex</code>	125
<code>mqc_algebra::mqc_cast_real</code>	125
<code>mqc_est::mqc_determinant</code>	126
<code>mqc_est::mqc_determinant_string</code>	127
<code>mqc_algebra::mqc_have_complex</code>	128
<code>mqc_algebra::mqc_have_int</code>	128
<code>mqc_algebra::mqc_have_real</code>	129
<code>mqc_algebra::mqc_matrix</code>	130
<code>mqc_algebra::mqc_matrix_diagmatrix_put</code>	134
<code>mqc_algebra::mqc_matrix_symmmatrix_put</code>	135
<code>mqc_est::mqc_matrix_undospinblockghf</code>	136

<code>mqc_algebra::mqc_print</code>	137
<code>mqc_est::mqc_print</code>	139
<code>mqc_algebra::mqc_r4tensor</code>	142
<code>mqc_algebra::mqc_scalar</code>	143
<code>mqc_est::mqc_scf_eigenvalues</code>	144
<code>mqc_est::mqc_scf_integral</code>	146
<code>mqc_algebra::mqc_set_array2vector</code>	149
<code>mqc_est::mqc_twoeris</code>	149
<code>mqc_algebra::mqc_vector</code>	150
<code>mqc_est::mqc_wavefunction</code>	155
<code>mqc_est::mqc_pscf_wavefunction</code>	140
<code>mqc_algebra::operator(*)</code>	159
<code>mqc_est::operator(*)</code>	163
<code>mqc_algebra::operator(**)</code>	164
<code>mqc_est::operator(+)</code>	167
<code>mqc_algebra::operator(+)</code>	168
<code>mqc_est::operator(-)</code>	170
<code>mqc_algebra::operator(-)</code>	171
<code>mqc_algebra::operator(.dot.)</code>	173
<code>mqc_algebra::operator(.eq.)</code>	174
<code>mqc_algebra::operator(.ewd.)</code>	175
<code>mqc_algebra::operator(.ewp.)</code>	176
<code>mqc_algebra::operator(.ge.)</code>	177
<code>mqc_algebra::operator(.gt.)</code>	177
<code>mqc_algebra::operator(.le.)</code>	180
<code>mqc_algebra::operator(.lt.)</code>	182
<code>mqc_algebra::operator(.ne.)</code>	184
<code>mqc_algebra::operator(.outer.)</code>	185
<code>mqc_algebra::operator(.x.)</code>	186
<code>mqc_algebra::operator(/)</code>	186
<code>mqc_algebra::real</code>	189
<code>mqc_algebra::sin</code>	190
<code>mqc_algebra::sqrt</code>	191
<code>mqc_algebra::tan</code>	192
<code>mqc_est::transpose</code>	193
<code>mqc_algebra::transpose</code>	193

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

mqc_algebra::abs	101
mqc_algebra::acos	102
mqc_algebra::aimag	103
mqc_algebra::asin	104
mqc_algebra::assignment(=)	105
mqc_est::assignment(=)	113
mqc_algebra::atan	113
mqc_algebra::atan2	114
mqc_algebra::cmplx	115
mqc_algebra::conjg	116
mqc_algebra::contraction	117
mqc_est::contraction	117
mqc_algebra::cos	118
mqc_algebra::dagger	119
mqc_est::dagger	120
mqc_algebra::dot_product	120
mqc_est::dot_product	121
mqc_algebra::matmul	121
mqc_est::matmul	122
mqc_algebra::matrix_symm2sq	124
mqc_algebra::mqc_cast_complex	125
mqc_algebra::mqc_cast_real	125
mqc_est::mqc_determinant	126
mqc_est::mqc_determinant_string	127
mqc_algebra::mqc_have_complex	128
mqc_algebra::mqc_have_int	128
mqc_algebra::mqc_have_real	129
mqc_algebra::mqc_matrix	130
mqc_algebra::mqc_matrix_diagmatrix_put	134
mqc_algebra::mqc_matrix_symmmatrix_put	135
mqc_est::mqc_matrix_undospinblockghf	136

mqc_algebra::mqc_print	137
mqc_est::mqc_print	139
mqc_est::mqc_pscf_wavefunction	140
mqc_algebra::mqc_r4tensor	142
mqc_algebra::mqc_scalar	143
mqc_est::mqc_scf_eigenvalues	144
mqc_est::mqc_scf_integral	146
mqc_algebra::mqc_set_array2vector	149
mqc_est::mqc_twoeris	149
mqc_algebra::mqc_vector	150
mqc_est::mqc_wavefunction	155
mqc_algebra::operator(*)	159
mqc_est::operator(*)	163
mqc_algebra::operator(**)	164
mqc_est::operator(+)	167
mqc_algebra::operator(+)	168
mqc_est::operator(-)	170
mqc_algebra::operator(-)	171
mqc_algebra::operator(.dot.)	173
mqc_algebra::operator(.eq.)	174
mqc_algebra::operator(.ewd.)	175
mqc_algebra::operator(.ewp.)	176
mqc_algebra::operator(.ge.)	177
mqc_algebra::operator(.gt.)	177
mqc_algebra::operator(.le.)	180
mqc_algebra::operator(.lt.)	182
mqc_algebra::operator(.ne.)	184
mqc_algebra::operator(.outer.)	185
mqc_algebra::operator(.x.)	186
mqc_algebra::operator(/)	186
mqc_algebra::real	189
mqc_algebra::sin	190
mqc_algebra::sqrt	191
mqc_algebra::tan	192
mqc_est::transpose	193
mqc_algebra::transpose	193

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/ mqc_algebra.F03	195
src/ mqc_est.F03	202

Chapter 5

Module Documentation

5.1 mqc_algebra Module Reference

Data Types

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

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

Functions/Subroutines

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

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- subroutine [mqc_scalar_get_random_value](#) (Scalar)

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

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

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

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

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

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

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

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

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

- type([mqc_scalar](#)) function [mqc_scalarexponent](#) (Scalar1, Scalar2)
***MQC_ScalarExponent** is a function that raises one MQC_Scalar to the power of another MQC_Scalar*
- type([mqc_scalar](#)) function [mqc_scalarintegerexponent](#) (Scalar, IntIn)
***MQC_ScalarIntegerExponent** is a function that raises an MQC_Scalar to the power of an intrinsic integer*
- type([mqc_scalar](#)) function [mqc_scalarrealexponent](#) (Scalar, 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*
- 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](#) (RealIn, Scalar)
***MQC_RealLTScalar** is a function that returns TRUE if an intrinsic real is less than a 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_scalargt](#) (Scalar1, Scalar2)
***MQC_ScalarGT** is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar*
- logical function [mqc_integertgtscalar](#) (IntIn, Scalar)
***MQC_IntegerGTScalar** is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar*
- logical function [mqc_scalargtinteger](#) (Scalar, IntIn)
***MQC_ScalarGTInteger** is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer*
- logical function [mqc_realgtscalar](#) (RealIn, Scalar)
- logical function [mqc_scalargtreal](#) (Scalar, RealIn)
- logical function [mqc_scalarle](#) (Scalar1, Scalar2)
- logical function [mqc_reallescalar](#) (RealIn, Scalar)
- logical function [mqc_scalarlereal](#) (Scalar, RealIn)
- logical function [mqc_integerlescalar](#) (IntIn, Scalar)
- logical function [mqc_scalarleinteger](#) (Scalar, IntIn)
- logical function [mqc_scalarge](#) (Scalar1, Scalar2)
- type([mqc_scalar](#)) function [mqc_scalar_complex_conjugate](#) (ScalarIn)
- type([mqc_scalar](#)) function [mqc_scalar_complex_realpart](#) (ScalarIn)
- type([mqc_scalar](#)) function [mqc_scalar_complex_imagpart](#) (ScalarIn)
- type([mqc_scalar](#)) function [mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
- type([mqc_scalar](#)) function [mqc_realscalarmultiply](#) (RealIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarrealmultiply](#) (Scalar, RealIn)
- type([mqc_scalar](#)) function [mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
- type([mqc_scalar](#)) function [mqc_integerscalardivide](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegerdivide](#) (Scalar, IntegerIn)
- type([mqc_scalar](#)) function [mqc_realscalardivide](#) (RealIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarrealdivide](#) (Scalar, RealIn)
- type([mqc_scalar](#)) function [mqc_complexscalardivide](#) (ComplexIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarcomplexdivide](#) (Scalar, ComplexIn)
- type([mqc_scalar](#)) function [mqc_integerscalaradd](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegeradd](#) (Scalar, IntegerIn)

- type(`mqc_scalar`) function `mqc_realscalaradd` (RealIn, Scalar)
- type(`mqc_scalar`) function `mqc_scalarrealadd` (Scalar, RealIn)
- type(`mqc_scalar`) function `mqc_complexscalaradd` (ComplexIn, Scalar)
- type(`mqc_scalar`) function `mqc_scalarcomplexadd` (Scalar, ComplexIn)
- type(`mqc_scalar`) function `mqc_integerscalarsubtract` (IntegerIn, Scalar)
- type(`mqc_scalar`) function `mqc_scalarintegersubtract` (Scalar, IntegerIn)
- type(`mqc_scalar`) function `mqc_realscalarsubtract` (RealIn, Scalar)
- type(`mqc_scalar`) function `mqc_scalarrealsubtract` (Scalar, RealIn)
- type(`mqc_scalar`) function `mqc_complexscalarsubtract` (ComplexIn, Scalar)
- type(`mqc_scalar`) function `mqc_scalarcomplexsubtract` (Scalar, ComplexIn)
- subroutine `mqc_allocate_vector` (N, Vector, Data_Type)
- subroutine `mqc_deallocate_vector` (Vector)
- integer(kind=int64) function `mqc_length_vector` (Vector)
- logical function `mqc_vector_havereal` (Vector)
- logical function `mqc_vector_haveinteger` (Vector)
- logical function `mqc_vector_havecomplex` (Vector)
- logical function `mqc_vector_iscolumn` (Vector)
- subroutine `mqc_vector_copy_int2real` (Vector)
- subroutine `mqc_vector_copy_int2complex` (Vector)
- subroutine `mqc_vector_copy_real2int` (Vector)
- subroutine `mqc_vector_copy_real2complex` (Vector)
- subroutine `mqc_vector_copy_complex2int` (Vector)
- subroutine `mqc_vector_copy_complex2real` (Vector)
- type(`mqc_scalar`) function `mqc_vector_scalar_at` (Vec, I)
- type(`mqc_vector`) function `mqc_vector_vector_at` (Vec, I, J)
- subroutine `mqc_set_vector2integerarray` (ArrayOut, VectorIn)
- subroutine `mqc_set_vector2realarray` (ArrayOut, VectorIn)
- subroutine `mqc_set_vector2complexarray` (ArrayOut, VectorIn)
- subroutine `mqc_set_array2vector_integer` (VectorOut, ArrayIn)
- subroutine `mqc_set_array2vector_real` (VectorOut, ArrayIn)
- subroutine `mqc_set_array2vector_complex` (VectorOut, ArrayIn)
- subroutine `mqc_set_vector2vector` (VectorOut, VectorIn)
- type(`mqc_vector`) function `mqc_vectorvectorsum` (Vector1In, Vector2In)
- type(`mqc_vector`) function `mqc_vectorvectordifference` (Vector1In, Vector2In)
- type(`mqc_vector`) function `mqc_scalarvectorsum` (ScalarIn, VectorIn)
- type(`mqc_vector`) function `mqc_scalarvectordifference` (ScalarIn, VectorIn)
- type(`mqc_vector`) function `mqc_elementvectorproduct` (Vector1In, Vector2In)
- type(`mqc_vector`) function `mqc_vector_transpose` (Vector)
- type(`mqc_vector`) function `mqc_vector_conjugate_transpose` (Vector)
- type(`mqc_scalar`) function `mqc_vectorvectordotproduct` (Vector1, Vector2)
- type(`mqc_matrix`) function `mqc_outer` (VA, VB)
- type(`mqc_vector`) function `mqc_crossproduct` (Vector1In, Vector2In)
- subroutine `mqc_print_vector_algebra1` (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- type(`mqc_vector`) function `mqc_vector_cast_real` (VA)
- type(`mqc_vector`) function `mqc_vector_cast_complex` (VA)
- subroutine `mqc_vector_scalar_put` (Vector, Scalar, I)
- subroutine `mqc_vector_scalar_increment` (Vector, Scalar, I)
- subroutine `mqc_vector_vector_put` (Vector, VectorIn, I)
- subroutine `mqc_vector_initialize` (Vector, Length, Scalar)
- type(`mqc_vector`) function `mqc_scalarvectorproduct` (Scalar, Vector)
- type(`mqc_vector`) function `mqc_vectorscalarproduct` (vector, scalar)

- type([mqc_vector](#)) function [mqc_vectorscalardivide](#) (vector, scalar)
- type([mqc_vector](#)) function [mqc_realvectorproduct](#) (Realln, Vector)
- type([mqc_vector](#)) function [mqc_vectorrealproduct](#) (vector, realln)
- type([mqc_vector](#)) function [mqc_vectorrealdive](#) (vector, realln)
- type([mqc_vector](#)) function [mqc_integervectorproduct](#) (intln, Vector)
- type([mqc_vector](#)) function [mqc_vectorintegerproduct](#) (vector, intln)
- type([mqc_vector](#)) function [mqc_vectorintegerdivide](#) (vector, intln)
- type([mqc_vector](#)) function [mqc_complexvectorproduct](#) (Compln, Vector)
- type([mqc_vector](#)) function [mqc_vectorcomplexproduct](#) (vector, compln)
- type([mqc_vector](#)) function [mqc_vectorcomplexdivide](#) (vector, compln)
- type([mqc_scalar](#)) function [mqc_vector_norm](#) (vector, methodln)
- logical function [mqc_vector_isallocated](#) (Vector)
- subroutine [mqc_vector_push](#) (Vector, Scalar)
- subroutine [mqc_vector_unshift](#) (Vector, Scalar)
- type([mqc_scalar](#)) function [mqc_vector_pop](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_shift](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_maxval](#) (Vector)
- type([mqc_scalar](#)) function [mqc_vector_minval](#) (Vector)
- integer function [mqc_vector_maxloc](#) (Vector)
- integer function [mqc_vector_minloc](#) (Vector)
- type([mqc_vector](#)) function [mqc_vector_argsort](#) (Vector)
- subroutine [mqc_vector_sort](#) (Vector, idx)
- subroutine [mqc_vector_sqrt](#) (A)
- type([mqc_vector](#)) function [mqc_vector_abs](#) (A)
- subroutine [mqc_vector_power](#) (A, P)
- type([mqc_vector](#)) function [mqc_vector_complex_realpart](#) (A)
- type([mqc_vector](#)) function [mqc_vector_complex_imagpart](#) (A)
- type([mqc_vector](#)) function [mqc_vector_cmplx](#) (Vector1, Vector2)
- character(len=64) function [mqc_matrix_storage_type](#) (Matrix)
- subroutine [mqc_matrix_diagonalize](#) (A, EVals, EVecs)
- type([mqc_matrix](#)) function [mqc_matrix_cast_real](#) (MA)
- type([mqc_matrix](#)) function [mqc_matrix_cast_complex](#) (MA)
- type([mqc_scalar](#)) function [mqc_matrix_scalar_at](#) (Mat, I, J)
- type([mqc_vector](#)) function [mqc_matrix_vector_at](#) (Mat, Rows, Cols)
- recursive subroutine [mqc_matrix_vector_put](#) (Mat, Vectorln, Rows, Cols)
- type([mqc_matrix](#)) function [mqc_matrix_matrix_at](#) (Mat, Rows, Cols)

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

- subroutine [mqc_matrix_diagmatrix_put_vector](#) (diagVectorln, mat)
- subroutine [mqc_matrix_diagmatrix_put_integer](#) (mat, diagMatrixln)
- subroutine [mqc_matrix_diagmatrix_put_real](#) (mat, diagMatrixln)
- subroutine [mqc_matrix_diagmatrix_put_complex](#) (mat, diagMatrixln)
- subroutine [mqc_matrix_symmmatrix_put_integer](#) (mat, symmMatrixln)
- subroutine [mqc_matrix_symmmatrix_put_real](#) (mat, symmMatrixln)
- subroutine [mqc_matrix_symmmatrix_put_complex](#) (mat, symmMatrixln)
- recursive subroutine [mqc_matrix_matrix_put](#) (Mat, Matrixln, Rows, Cols)
- integer(kind=int64) function [symindexhash](#) (i, j, k, l)
- type([mqc_matrix](#)) function [mqc_elementmatrixproduct](#) (A, B)
- type([mqc_matrix](#)) function [mqc_elementmatrixdivide](#) (A, B)
- logical function [mqc_matrix_test_symmetric](#) (Matrix, Option)
- logical function [mqc_matrix_test_diagonal](#) (Matrix)

- subroutine [mqc_allocate_matrix](#) (M, N, Matrix, Data_Type, Storage)
- subroutine [mqc_deallocate_matrix](#) (Matrix)
- logical function [mqc_matrix_isallocated](#) (Matrix)
- subroutine [mqc_set_integerarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_realarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_complexarray2matrix](#) (MatrixOut, ArrayIn)
- subroutine [mqc_set_matrix2integerarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2realarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2complexarray](#) (ArrayOut, MatrixIn)
- subroutine [mqc_set_matrix2matrix](#) (MatrixOut, MatrixIn)
- subroutine [mqc_print_matrix_algebra1](#) (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_matrix_copy_int2real](#) (Matrix)
- subroutine [mqc_matrix_copy_int2complex](#) (Matrix)
- subroutine [mqc_matrix_copy_real2int](#) (Matrix)
- subroutine [mqc_matrix_copy_real2complex](#) (Matrix)
- subroutine [mqc_matrix_copy_complex2int](#) (Matrix)
- subroutine [mqc_matrix_copy_complex2real](#) (Matrix)
- integer(kind=int64) function [mqc_matrix_rows](#) (Matrix)
- integer(kind=int64) function [mqc_matrix_columns](#) (Matrix)
- logical function [mqc_matrix_havereal](#) (Matrix)
- logical function [mqc_matrix_haveinteger](#) (Matrix)
- logical function [mqc_matrix_havecomplex](#) (Matrix)
- logical function [mqc_matrix_havfull](#) (Matrix)
- logical function [mqc_matrix_havesymmetric](#) (Matrix)
- logical function [mqc_matrix_havediagonal](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_transpose](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_conjugate_transpose](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_symmetrize](#) (Matrix)
- subroutine [mqc_matrix_full2symm](#) (Matrix)
- subroutine [mqc_matrix_symm2full](#) (Matrix, Option)
- subroutine [mqc_matrix_full2diag](#) (Matrix)
- subroutine [mqc_matrix_diag2full](#) (Matrix)
- subroutine [mqc_matrix_symm2diag](#) (Matrix)
- subroutine [mqc_matrix_diag2symm](#) (Matrix)
- type([mqc_matrix](#)) function [mqc_matrix_symm2full_func](#) (Matrix)
- subroutine [matrix_symm2sq_integer](#) (N, I_Symm, I_Sq)
- subroutine [matrix_symm2sq_real](#) (N, A_Symm, A_Sq)
- subroutine [matrix_symm2sq_complex](#) (N, A_Symm, A_Sq)
- type([mqc_matrix](#)) function [mqc_vector2diagmatrix](#) (vector)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsum](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsubtract](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixproduct](#) (MA, MB)
- type([mqc_matrix](#)) function [mqc_matrixmatrixdotproduct](#) (MA, MB)
- type([mqc_vector](#)) function [mqc_matrixvectordotproduct](#) (MA, VB)
- type([mqc_vector](#)) function [mqc_vectormatrixdotproduct](#) (VA, MB)
- type([mqc_matrix](#)) function [mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type([mqc_matrix](#)) function [mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type([mqc_scalar](#)) function [mqc_matrix_matrix_contraction](#) (Matrix1, Matrix2)
- subroutine [mqc_matrix_scalar_put](#) (Matrix, Scalar, I, J)
- subroutine [mqc_matrix_initialize](#) (Matrix, Rows, Columns, Scalar, Storage)
- subroutine [mqc_matrix_identity](#) (matrix, n, m)

- subroutine `mqc_matrix_set` (matrix, scalar, storage)
- type(`mqc_scalar`) function `mqc_matrix_norm` (matrix, methodIn)
- type(`mqc_scalar`) function `mqc_matrix_determinant` (a)
- type(`mqc_matrix`) function `mqc_matrix_inverse` (a)
- type(`mqc_scalar`) function `mqc_matrix_trace` (matrix)
- subroutine `mqc_matrix_generalized_eigensystem` (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine `mqc_matrix_svd` (A, EVals, EUVecs, EVVecs)
- subroutine `mqc_matrix_rms_max` (A, rms_A, max_A)
- subroutine `mqc_matrix_sqrt` (A, eVals, eVecs)
- type(`mqc_matrix`) function `mqc_givens_matrix` (m_size, angle, p, q)
- subroutine `mqc_allocate_r4tensor` (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine `mqc_deallocate_r4tensor` (Tensor)
- type(`mqc_scalar`) function `mqc_r4tensor_at` (Tensor, I, J, K, L)
- subroutine `mqc_r4tensor_put` (Tensor, Element, I, J, K, L)
- subroutine `mqc_print_r4tensor_algebra1` (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine `mqc_set_array2tensor` (TensorOut, ArrayIn)
- subroutine `mqc_r4tensor_initialize` (R4Tensor, I, J, K, L, Scalar)
- subroutine `mqc_matrix_symmsymmr4tensor_put_real` (r4Tensor, symmSymmMatrixIn)
- subroutine `mqc_matrix_symmsymmr4tensor_put_complex` (r4Tensor, symmSymmMatrixIn)
- logical function `mqc_r4tensor_haveinteger` (R4Tensor)
- logical function `mqc_r4tensor_havereal` (R4Tensor)
- logical function `mqc_r4tensor_havecomplex` (R4Tensor)

5.1.1 Function/Subroutine Documentation

5.1.1.1 `bin_coeff()`

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

Bin_Coeff returns the binomial coefficient of (n,k)

Purpose:

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

Parameters

in	<i>N</i>	N is Integer(kind=int64) The number of objects
in	<i>K</i>	K is Integer(kind=int64) The number of permutations

Author

L. M. Thompson

Date

2016

5.1.1.2 factorial()

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

Factorial returns the factorial of an integer**Purpose:**

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.3 matrix_symm2sq_complex()

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

5.1.1.4 matrix_symm2sq_integer()

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

5.1.1.5 matrix_symm2sq_real()

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

5.1.1.6 mqc_allocate_matrix()

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

5.1.1.7 mqc_allocate_r4tensor()

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


5.1.1.8 mqc_allocate_scalar()

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

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

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.9 mqc_allocate_vector()

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

5.1.1.10 mqc_complexscalaradd()

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

5.1.1.11 mqc_complexscalardivide()

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

5.1.1.12 mqc_complexscalarmultiply()

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

5.1.1.13 mqc_complexscalarsubtract()

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

5.1.1.14 mqc_complexvectorproduct()

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

5.1.1.15 mqc_crossproduct()

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

5.1.1.16 mqc_deallocate_matrix()

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

5.1.1.17 mqc_deallocate_r4tensor()

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

5.1.1.18 mqc_deallocate_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.19 mqc_deallocate_vector()

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

5.1.1.20 mqc_elementmatrixdivide()

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

5.1.1.21 mqc_elementmatrixproduct()

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

5.1.1.22 mqc_elementvectorproduct()

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

5.1.1.23 mqc_givens_matrix()

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

5.1.1.24 mqc_input_complex_scalar()

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

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

Purpose:

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

Parameters

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

5.1.1.25 mqc_input_integer_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.26 mqc_input_real_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.27 mqc_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. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.28 mqc_integerlescalar()

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

5.1.1.29 mqc_integerscalaradd()

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

5.1.1.30 mqc_integerscalardivide()

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

5.1.1.31 mqc_integerscalarmultiply()

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

5.1.1.32 mqc_integerscalarsubtract()

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

5.1.1.33 mqc_integervectorproduct()

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

5.1.1.34 mqc_length_vector()

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

5.1.1.35 mqc_matrix_cast_complex()

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


5.1.1.36 mqc_matrix_cast_real()

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

5.1.1.37 mqc_matrix_columns()

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

5.1.1.38 mqc_matrix_conjugate_transpose()

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

5.1.1.39 mqc_matrix_copy_complex2int()

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

5.1.1.40 mqc_matrix_copy_complex2real()

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

5.1.1.41 mqc_matrix_copy_int2complex()

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

5.1.1.42 mqc_matrix_copy_int2real()

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

5.1.1.43 mqc_matrix_copy_real2complex()

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

5.1.1.44 mqc_matrix_copy_real2int()

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

5.1.1.45 mqc_matrix_determinant()

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

5.1.1.46 mqc_matrix_diag2full()

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

5.1.1.47 mqc_matrix_diag2symm()

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

5.1.1.48 mqc_matrix_diagmatrix_put_complex()

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

5.1.1.49 mqc_matrix_diagmatrix_put_integer()

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

5.1.1.50 mqc_matrix_diagmatrix_put_real()

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

5.1.1.51 mqc_matrix_diagmatrix_put_vector()

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

5.1.1.52 mqc_matrix_diagonalize()

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

5.1.1.53 mqc_matrix_full2diag()

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

5.1.1.54 mqc_matrix_full2symm()

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

5.1.1.55 mqc_matrix_generalized_eigensystem()

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

5.1.1.56 mqc_matrix_havecomplex()

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

5.1.1.57 mqc_matrix_havediagonal()

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

5.1.1.58 mqc_matrix_havefull()

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

5.1.1.59 mqc_matrix_haveinteger()

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

5.1.1.60 mqc_matrix_havereal()

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

5.1.1.61 mqc_matrix_havesymmetric()

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

5.1.1.62 mqc_matrix_identity()

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

5.1.1.63 mqc_matrix_initialize()

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

5.1.1.64 mqc_matrix_inverse()

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

5.1.1.65 mqc_matrix_isallocated()

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

5.1.1.66 `mqc_matrix_matrix_at()`

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

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.67 mqc_matrix_matrix_contraction()

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

5.1.1.68 mqc_matrix_matrix_put()

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

5.1.1.69 mqc_matrix_norm()

```

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

```

5.1.1.70 mqc_matrix_rms_max()

```

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

```

5.1.1.71 mqc_matrix_rows()

```

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

```

5.1.1.72 mqc_matrix_scalar_at()

```

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

```

5.1.1.73 mqc_matrix_scalar_put()

```

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

```


5.1.1.74 mqc_matrix_set()

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

5.1.1.75 mqc_matrix_sqrt()

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

5.1.1.76 mqc_matrix_storage_type()

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

5.1.1.77 mqc_matrix_svd()

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

5.1.1.78 mqc_matrix_symm2diag()

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

5.1.1.79 mqc_matrix_symm2full()

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

5.1.1.80 mqc_matrix_symm2full_func()

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

5.1.1.81 mqc_matrix_symmetrize()

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

5.1.1.82 mqc_matrix_symmmatrix_put_complex()

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

5.1.1.83 mqc_matrix_symmmatrix_put_integer()

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

5.1.1.84 mqc_matrix_symmmatrix_put_real()

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

5.1.1.85 mqc_matrix_symmsymmr4tensor_put_complex()

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

5.1.1.86 mqc_matrix_symmsymmr4tensor_put_real()

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

5.1.1.87 mqc_matrix_test_diagonal()

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

5.1.1.88 mqc_matrix_test_symmetric()

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

5.1.1.89 mqc_matrix_trace()

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

5.1.1.90 mqc_matrix_transpose()

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

5.1.1.91 mqc_matrix_vector_at()

```

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

```

5.1.1.92 mqc_matrix_vector_put()

```

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

```

5.1.1.93 mqc_matrixmatrixdotproduct()

```

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

```

5.1.1.94 mqc_matrixmatrixproduct()

```

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

```

5.1.1.95 mqc_matrixmatrixsubtract()

```

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

```

5.1.1.96 mqc_matrixmatrixsum()

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

5.1.1.97 mqc_matrixscalarproduct()

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

5.1.1.98 mqc_matrixvectordotproduct()

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

5.1.1.99 mqc_outer()

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

5.1.1.100 mqc_output_complex_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.101 mqc_output_integer_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.102 mqc_output_mqcscalar_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.103 mqc_output_real_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.104 mqc_print_matrix_algebra1()

```

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

```

5.1.1.105 mqc_print_r4tensor_algebra1()

```

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

```


5.1.1.106 mqc_print_scalar_algebra1()

```

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

```

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.107 mqc_print_vector_algebra1()

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

5.1.1.108 mqc_r4tensor_at()

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

5.1.1.109 mqc_r4tensor_havecomplex()

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

5.1.1.110 mqc_r4tensor_haveinteger()

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

5.1.1.111 mqc_r4tensor_havereal()

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

5.1.1.112 mqc_r4tensor_initialize()

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

5.1.1.113 mqc_r4tensor_put()

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

5.1.1.114 mqc_realgtscalar()

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

5.1.1.115 mqc_reallescalar()

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

5.1.1.116 `mqc_realltscalar()`

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar. If the intrinsic real is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.117 `mqc_realscalaradd()`

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

5.1.1.118 mqc_realscalardivide()

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

5.1.1.119 mqc_realscalarmultiply()

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

5.1.1.120 mqc_realscalarsubtract()

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

5.1.1.121 mqc_realvectorproduct()

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

5.1.1.122 mqc_scalar_acos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.123 mqc_scalar_asin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.124 mqc_scalar_atan()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.125 mqc_scalar_atan2()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.126 mqc_scalar_cmplx()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.127 mqc_scalar_complex_conjugate()

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


5.1.1.128 mqc_scalar_complex_imagpart()

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

5.1.1.129 mqc_scalar_complex_realpart()

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

5.1.1.130 mqc_scalar_cos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.131 mqc_scalar_get_abs_value()

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

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

Purpose:

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

Parameters

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

Author

A. Mahler

Date

2018

5.1.1.132 mqc_scalar_get_intrinsic_complex()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.133 mqc_scalar_get_intrinsic_integer()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.134 mqc_scalar_get_intrinsic_real()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.135 mqc_scalar_get_random_value()

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

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

Purpose:

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

Parameters

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

Author

X. Dong

Date

2019

5.1.1.136 mqc_scalar_havecomplex()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.137 mqc_scalar_haveinteger()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.138 mqc_scalar_havereal()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.139 mqc_scalar_isallocated()

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

MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.1.140 mqc_scalar_sin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.141 mqc_scalar_sqrt()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.142 mqc_scalar_tan()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.143 mqc_scalaradd()

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

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

Purpose:

MQC_ScalarAdd is a function that sums two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.144 mqc_scalarcomplexadd()

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

5.1.1.145 mqc_scalarcomplexdivide()

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

5.1.1.146 mqc_scalarcomplexexponent()

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

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

Purpose:

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

Parameters

in	<i>Scalar</i>	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

5.1.1.147 mqc_scalarcomplexmultiply()

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

5.1.1.148 mqc_scalarcomplexsubtract()

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

5.1.1.149 mqc_scalardivide()

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

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

Purpose:

MQC_ScalarDivide is a function that divides MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.150 mqc_scalareq()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.151 mqc_scalarexponent()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.152 mqc_scalarge()

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

5.1.1.153 mqc_scalargt()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the left real part is greater than the right real part and FALSE if the left real part is less than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is greater than the right imaginary part and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.154 mqc_scalargtinteger()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is greater than the intrinsic integer and FALSE if the real part of the MQC_Scalar is less than the intrinsic integer. If the real part of the MQC_Scalar is equal to the intrinsic integer, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.155 mqc_scalargtreal()

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

5.1.1.156 mqc_scalarintegeradd()

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

5.1.1.157 mqc_scalarintegerdivide()

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

5.1.1.158 mqc_scalarintegerexponent()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.159 mqc_scalarintegermultiply()

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

5.1.1.160 mqc_scalarintegersubtract()

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

5.1.1.161 mqc_scalarle()

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


5.1.1.162 mqc_scalarleinteger()

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

5.1.1.163 mqc_scalarlereal()

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

5.1.1.164 mqc_scalarlt()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the left real part is less than the right real part and FALSE if the left real part is greater than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is less than the right imaginary part and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.165 mqc_scalarltreal()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the real part of the MQC_Scalar is less than the intrinsic real and FALSE if the real part of the MQC_Scalar is greater than the intrinsic real. If the real part of the MQC_Scalar is equal to the intrinsic real, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.166 mqc_scalarmatrixproduct()

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

5.1.1.167 mqc_scalarmultiply()

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

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

Purpose:

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.168 mqc_scalarne()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.169 mqc_scalarrealadd()

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

5.1.1.170 mqc_scalarrealdive()

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

5.1.1.171 mqc_scalarrealexponent()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.1.172 mqc_scalarrealmultiply()

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

5.1.1.173 mqc_scalarrealsubtract()

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

5.1.1.174 mqc_scalarsubtract()

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

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

Purpose:

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.1.175 mqc_scalarvectordifference()

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

5.1.1.176 mqc_scalarvectorproduct()

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

5.1.1.177 mqc_scalarvectorsum()

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

5.1.1.178 mqc_set_array2tensor()

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

5.1.1.179 mqc_set_array2vector_complex()

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

5.1.1.180 mqc_set_array2vector_integer()

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

5.1.1.181 mqc_set_array2vector_real()

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

5.1.1.182 mqc_set_complexarray2matrix()

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

5.1.1.183 mqc_set_integerarray2matrix()

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

5.1.1.184 mqc_set_matrix2complexarray()

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

5.1.1.185 mqc_set_matrix2integerarray()

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

5.1.1.186 mqc_set_matrix2matrix()

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

5.1.1.187 mqc_set_matrix2realarray()

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

5.1.1.188 mqc_set_realarray2matrix()

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

5.1.1.189 mqc_set_vector2complexarray()

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


5.1.1.190 mqc_set_vector2integerarray()

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

5.1.1.191 mqc_set_vector2realarray()

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

5.1.1.192 mqc_set_vector2vector()

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

5.1.1.193 mqc_vector2diagmatrix()

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

5.1.1.194 mqc_vector_abs()

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

5.1.1.195 mqc_vector_argsort()

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

5.1.1.196 mqc_vector_cast_complex()

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

5.1.1.197 mqc_vector_cast_real()

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

5.1.1.198 mqc_vector_cmplx()

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

5.1.1.199 mqc_vector_complex_imagpart()

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

5.1.1.200 mqc_vector_complex_realpart()

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

5.1.1.201 mqc_vector_conjugate_transpose()

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

5.1.1.202 mqc_vector_copy_complex2int()

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

5.1.1.203 mqc_vector_copy_complex2real()

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

5.1.1.204 mqc_vector_copy_int2complex()

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

5.1.1.205 mqc_vector_copy_int2real()

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

5.1.1.206 mqc_vector_copy_real2complex()

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

5.1.1.207 mqc_vector_copy_real2int()

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

5.1.1.208 mqc_vector_havecomplex()

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

5.1.1.209 mqc_vector_haveinteger()

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

5.1.1.210 mqc_vector_havereal()

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

5.1.1.211 mqc_vector_initialize()

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

5.1.1.212 mqc_vector_isallocated()

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

5.1.1.213 mqc_vector_iscolumn()

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

5.1.1.214 mqc_vector_maxloc()

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

5.1.1.215 mqc_vector_maxval()

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

5.1.1.216 mqc_vector_minloc()

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

5.1.1.217 mqc_vector_minval()

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

5.1.1.218 mqc_vector_norm()

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

5.1.1.219 mqc_vector_pop()

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

5.1.1.220 mqc_vector_power()

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

5.1.1.221 mqc_vector_push()

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

5.1.1.222 mqc_vector_scalar_at()

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

5.1.1.223 mqc_vector_scalar_increment()

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

5.1.1.224 mqc_vector_scalar_put()

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

5.1.1.225 mqc_vector_shift()

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

5.1.1.226 mqc_vector_sort()

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

5.1.1.227 mqc_vector_sqrt()

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

5.1.1.228 mqc_vector_transpose()

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

5.1.1.229 mqc_vector_unshift()

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

5.1.1.230 mqc_vector_vector_at()

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

5.1.1.231 mqc_vector_vector_put()

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

5.1.1.232 mqc_vectorcomplexdivide()

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

5.1.1.233 mqc_vectorcomplexproduct()

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

5.1.1.234 mqc_vectorintegerdivide()

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

5.1.1.235 mqc_vectorintegerproduct()

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

5.1.1.236 mqc_vectormatrixdotproduct()

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

5.1.1.237 mqc_vectorrealddivide()

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


5.1.1.238 mqc_vectorrealproduct()

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

5.1.1.239 mqc_vectorscalardivide()

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

5.1.1.240 mqc_vectorscalarproduct()

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

5.1.1.241 mqc_vectorvectordifference()

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

5.1.1.242 mqc_vectorvectordotproduct()

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

5.1.1.243 mqc_vectorvectorsum()

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

5.1.1.244 symindexhash()

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

5.2 mqc_est Module Reference

Data Types

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

Functions/Subroutines

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

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

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

5.2.1 Function/Subroutine Documentation

5.2.1.1 gen_det_str()

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

5.2.1.2 get_one_gamma_matrix()

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

5.2.1.3 mqc_build_ci_hamiltonian()

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

5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()

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

5.2.1.5 mqc_eigenvalues_add_name()

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

5.2.1.6 mqc_eigenvalues_allocate()

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

5.2.1.7 mqc_eigenvalues_array_name()

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

5.2.1.8 mqc_eigenvalues_array_type()

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

5.2.1.9 mqc_eigenvalues_at()

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

5.2.1.10 mqc_eigenvalues_dimension()

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

5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()

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

5.2.1.12 mqc_eigenvalues_has_alpha()

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

5.2.1.13 mqc_eigenvalues_has_beta()

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

5.2.1.14 mqc_eigenvalues_integral_multiply()

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

5.2.1.15 mqc_eigenvalues_isallocated()

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

5.2.1.16 mqc_eigenvalues_output_array()

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

5.2.1.17 mqc_eigenvalues_output_block()

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

5.2.1.18 mqc_eri_integral_contraction()

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

5.2.1.19 mqc_integral_add_name()

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

5.2.1.20 mqc_integral_allocate()

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

5.2.1.21 mqc_integral_array_name()

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

5.2.1.22 mqc_integral_array_type()

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

5.2.1.23 mqc_integral_at()

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

5.2.1.24 mqc_integral_conjugate_transpose()

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

5.2.1.25 mqc_integral_delete_energy_list()

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

5.2.1.26 mqc_integral_difference()

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


5.2.1.27 mqc_integral_dimension()

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

5.2.1.28 mqc_integral_eigenvalues_multiply()

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

5.2.1.29 mqc_integral_get_energy_list()

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

5.2.1.30 mqc_integral_has_alpha()

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

5.2.1.31 mqc_integral_has_alphabeta()

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

5.2.1.32 mqc_integral_has_beta()

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

5.2.1.33 mqc_integral_has_betaalpha()

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

5.2.1.34 mqc_integral_identity()

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

5.2.1.35 mqc_integral_initialize()

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

5.2.1.36 mqc_integral_integral_multiply()

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

5.2.1.37 mqc_integral_isallocated()

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

5.2.1.38 mqc_integral_matrix_multiply()

```

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

```

5.2.1.39 mqc_integral_norm()

```

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

```

5.2.1.40 mqc_integral_output_array()

```

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

```

5.2.1.41 mqc_integral_output_block()

```

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

```

5.2.1.42 mqc_integral_output_orbitals()

```

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

```

5.2.1.43 mqc_integral_scalar_multiply()

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

5.2.1.44 mqc_integral_set_energy_list()

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

5.2.1.45 mqc_integral_sum()

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

5.2.1.46 mqc_integral_swap_orbitals()

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

5.2.1.47 mqc_integral_transpose()

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

5.2.1.48 mqc_matrix_integral_multiply()

```

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

```

5.2.1.49 mqc_matrix_spinblockghf()

```

subroutine mqc_est::mqc_matrix_spinblockghf (
    class(*), intent(inout) array,
    integer(kind=int64), optional 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

Public Member Functions

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

6.1.1 Member Function/Subroutine Documentation

6.1.1.1 mqc_scalar_get_abs_value()

```
type(mqc\_scalar) function mqc_algebra::abs::mqc_scalar_get_abs_value (  
    class(mqc\_scalar), intent(in) Scalar )
```

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

Purpose:

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

Parameters

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

Author

A. Mahler

Date

2018

6.1.1.2 mqc_vector_abs()

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

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

- src/mqc_algebra.F03

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

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

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

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/[mqc_algebra.F03](#)

6.3 mqc_algebra::aimag Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalar_complex_imagpart](#) (ScalarIn)
- type([mqc_vector](#)) function [mqc_vector_complex_imagpart](#) (A)

6.3.1 Member Function/Subroutine Documentation

6.3.1.1 mqc_scalar_complex_imagpart()

```
type(mqc\_scalar) function mqc_algebra::aimag::mqc_scalar_complex_imagpart (
    type(mqc\_scalar), intent(in) ScalarIn )
```

6.3.1.2 mqc_vector_complex_imagpart()

```
type(mqc\_vector) function mqc_algebra::aimag::mqc_vector_complex_imagpart (
    class(mqc\_vector), intent(in) A )
```

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

- src/[mqc_algebra.F03](#)

6.4 mqc_algebra::asin Interface Reference

Public Member Functions

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

6.4.1 Member Function/Subroutine Documentation

6.4.1.1 mqc_scalar_asin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- `src/mqc_algebra.F03`

6.5 mqc_algebra::assignment(=) Interface Reference

Public Member Functions

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

6.5.1 Member Function/Subroutine Documentation

6.5.1.1 mqc_input_complex_scalar()

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

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

Purpose:

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

Parameters

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

6.5.1.2 mqc_input_integer_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.1.3 mqc_input_real_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.1.4 mqc_output_complex_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.1.5 mqc_output_integer_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.1.6 mqc_output_mqcscalar_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.1.7 mqc_output_real_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.1.8 mqc_set_array2tensor()

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

6.5.1.9 mqc_set_array2vector_complex()

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

6.5.1.10 mqc_set_array2vector_integer()

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

6.5.1.11 mqc_set_array2vector_real()

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

6.5.1.12 mqc_set_complexarray2matrix()

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

6.5.1.13 mqc_set_integerarray2matrix()

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

6.5.1.14 mqc_set_matrix2complexarray()

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

6.5.1.15 mqc_set_matrix2integerarray()

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

6.5.1.16 mqc_set_matrix2matrix()

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

6.5.1.17 mqc_set_matrix2realarray()

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

6.5.1.18 mqc_set_realarray2matrix()

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

6.5.1.19 mqc_set_vector2complexarray()

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

6.5.1.20 mqc_set_vector2integerarray()

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

6.5.1.21 mqc_set_vector2realarray()

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

6.5.1.22 mqc_set_vector2vector()

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

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

- src/[mqc_algebra.F03](#)

6.6 mqc_est::assignment(=) Interface Reference

Public Member Functions

- subroutine [mqc_integral_output_array](#) (matrixOut, 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

Public Member Functions

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

6.7.1 Member Function/Subroutine Documentation

6.7.1.1 mqc_scalar_atan()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.8 mqc_algebra::atan2 Interface Reference

Public Member Functions

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

6.8.1 Member Function/Subroutine Documentation

6.8.1.1 mqc_scalar_atan2()

```
type(mqc\_scalar) function mqc_algebra::atan2::mqc_scalar_atan2 (
    type(mqc\_scalar), intent(in) Scalar )
```

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.9 mqc_algebra::cmplx Interface Reference

Public Member Functions

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

6.9.1 Member Function/Subroutine Documentation

6.9.1.1 mqc_scalar_cmplx()

```
type(mqc\_scalar) function mqc_algebra::cmplx::mqc_scalar_cmplx (
    type(mqc\_scalar), intent(in) Scalar1,
    type(mqc\_scalar), intent(in) Scalar2 )
```

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

6.9.1.2 mqc_vector_cmplx()

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

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

- src/[mqc_algebra.F03](#)

6.10 mqc_algebra::conjg Interface Reference**Public Member Functions**

- type([mqc_scalar](#)) function [mqc_scalar_complex_conjugate](#) (ScalarIn)

6.10.1 Member Function/Subroutine Documentation

6.10.1.1 mqc_scalar_complex_conjugate()

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

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

- [src/mqc_algebra.F03](#)

6.11 mqc_algebra::contraction Interface Reference

Public Member Functions

- [type\(mqc_scalar\) function mqc_matrix_matrix_contraction](#) (Matrix1, Matrix2)

6.11.1 Member Function/Subroutine Documentation

6.11.1.1 mqc_matrix_matrix_contraction()

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

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

- [src/mqc_algebra.F03](#)

6.12 mqc_est::contraction Interface Reference

Public Member Functions

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

6.12.1 Member Function/Subroutine Documentation

6.12.1.1 mqc_eri_integral_contraction()

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

6.12.1.2 mqc_scf_integral_contraction()

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

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

- [src/mqc_est.F03](#)

6.13 mqc_algebra::cos Interface Reference

Public Member Functions

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

6.13.1 Member Function/Subroutine Documentation

6.13.1.1 mqc_scalar_cos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/[mqc_algebra.F03](#)

6.14 mqc_algebra::dagger Interface Reference

Public Member Functions

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

6.14.1 Member Function/Subroutine Documentation

6.14.1.1 mqc_matrix_conjugate_transpose()

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

6.14.1.2 mqc_vector_conjugate_transpose()

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

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

- src/[mqc_algebra.F03](#)

6.15 mqc_est::dagger Interface Reference

Public Member Functions

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

6.15.1 Member Function/Subroutine Documentation

6.15.1.1 mqc_integral_conjugate_transpose()

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

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

- src/[mqc_est.F03](#)

6.16 mqc_algebra::dot_product Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_vectorvectordotproduct](#) (Vector1, Vector2)

6.16.1 Member Function/Subroutine Documentation

6.16.1.1 mqc_vectorvectordotproduct()

```
type(mqc\_scalar) function mqc_algebra::dot_product::mqc_vectorvectordotproduct (
    type(mqc\_vector), intent(in) Vector1,
    type(mqc\_vector), intent(in) Vector2 )
```

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

- src/[mqc_algebra.F03](#)

6.17 `mqc_est::dot_product` Interface Reference

Public Member Functions

- `type(mqc_scalar)` function [mqc_eigenvalue_eigenvalue_dotproduct](#) (`eigenvalueA`, `eigenvalueB`)

6.17.1 Member Function/Subroutine Documentation

6.17.1.1 `mqc_eigenvalue_eigenvalue_dotproduct()`

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

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

- `src/mqc_est.F03`

6.18 `mqc_algebra::matmul` Interface Reference

Public Member Functions

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

6.18.1 Member Function/Subroutine Documentation

6.18.1.1 `mqc_matrixmatrixdotproduct()`

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

6.18.1.2 mqc_matrixvectordotproduct()

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

6.18.1.3 mqc_vectormatrixdotproduct()

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

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

- src/[mqc_algebra.F03](#)

6.19 mqc_est::matmul Interface Reference

Public Member Functions

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

6.19.1 Member Function/Subroutine Documentation

6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()

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


6.19.1.2 mqc_eigenvalues_integral_multiply()

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

6.19.1.3 mqc_integral_eigenvalues_multiply()

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

6.19.1.4 mqc_integral_integral_multiply()

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

6.19.1.5 mqc_integral_matrix_multiply()

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

6.19.1.6 mqc_matrix_integral_multiply()

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

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

- [src/mqc_est.F03](#)

6.20 mqc_algebra::matrix_symm2sq Interface Reference

Public Member Functions

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

6.20.1 Member Function/Subroutine Documentation

6.20.1.1 [matrix_symm2sq_complex\(\)](#)

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

6.20.1.2 [matrix_symm2sq_integer\(\)](#)

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

6.20.1.3 [matrix_symm2sq_real\(\)](#)

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

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

- [src/mqc_algebra.F03](#)

6.21 mqc_algebra::mqc_cast_complex Interface Reference

Public Member Functions

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

6.21.1 Member Function/Subroutine Documentation

6.21.1.1 mqc_matrix_cast_complex()

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

6.21.1.2 mqc_vector_cast_complex()

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

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

- src/[mqc_algebra.F03](#)

6.22 mqc_algebra::mqc_cast_real Interface Reference

Public Member Functions

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

6.22.1 Member Function/Subroutine Documentation

6.22.1.1 `mqc_matrix_cast_real()`

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

6.22.1.2 `mqc_vector_cast_real()`

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

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

- [src/mqc_algebra.F03](#)

6.23 `mqc_est::mqc_determinant` Type Reference

Public Attributes

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

6.23.1 Member Data Documentation

6.23.1.1 `nalpstr`

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

6.23.1.2 `nbetstr`

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

6.23.1.3 ndets

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

6.23.1.4 order

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

6.23.1.5 strings

```
type(mqc_determinant_string) mqc_est::mqc_determinant::strings
```

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

- [src/mqc_est.F03](#)

6.24 mqc_est::mqc_determinant_string Type Reference

Public Attributes

- type(mqc_matrix) [alpha](#)
- type(mqc_matrix) [beta](#)

6.24.1 Member Data Documentation

6.24.1.1 alpha

```
type(mqc_matrix) mqc_est::mqc_determinant_string::alpha
```

6.24.1.2 beta

```
type(mqc_matrix) mqc_est::mqc_determinant_string::beta
```

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

- [src/mqc_est.F03](#)

6.25 mqc_algebra::mqc_have_complex Interface Reference

Public Member Functions

- logical function [mqc_vector_havecomplex](#) (Vector)
- logical function [mqc_matrix_havecomplex](#) (Matrix)

6.25.1 Member Function/Subroutine Documentation

6.25.1.1 mqc_matrix_havecomplex()

```
logical function mqc_algebra::mqc_have_complex::mqc_matrix_havecomplex (  
    type(mqc\_matrix), intent(in) Matrix )
```

6.25.1.2 mqc_vector_havecomplex()

```
logical function mqc_algebra::mqc_have_complex::mqc_vector_havecomplex (  
    type(mqc\_vector), intent(in) Vector )
```

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

- src/[mqc_algebra.F03](#)

6.26 mqc_algebra::mqc_have_int Interface Reference

Public Member Functions

- logical function [mqc_vector_haveinteger](#) (Vector)
- logical function [mqc_matrix_haveinteger](#) (Matrix)

6.26.1 Member Function/Subroutine Documentation

6.26.1.1 mqc_matrix_haveinteger()

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

6.26.1.2 mqc_vector_haveinteger()

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

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

- [src/mqc_algebra.F03](#)

6.27 mqc_algebra::mqc_have_real Interface Reference

Public Member Functions

- logical function [mqc_vector_havereal](#) (Vector)
- logical function [mqc_matrix_havereal](#) (Matrix)

6.27.1 Member Function/Subroutine Documentation

6.27.1.1 mqc_matrix_havereal()

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

6.27.1.2 mqc_vector_havereal()

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

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

- [src/mqc_algebra.F03](#)

6.28 mqc_algebra::mqc_matrix Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_matrix_algebra1](#)
- Procedure, public [initialize](#) => [mqc_matrix_initialize](#)
- Procedure, public [init](#) => [mqc_matrix_initialize](#)
- Procedure, public [identity](#) => [mqc_matrix_identity](#)
- Procedure, public [set](#) => [mqc_matrix_set](#)
- Procedure, public [norm](#) => [mqc_matrix_norm](#)
- Procedure, public [transpose](#) => [mqc_matrix_transpose](#)
- Procedure, public [dagger](#) => [mqc_matrix_conjugate_transpose](#)
- Procedure, public [diag](#) => [mqc_matrix_diagonalize](#)
- Procedure, public [svd](#) => [mqc_matrix_svd](#)
- Procedure, public [eigensys](#) => [mqc_matrix_generalized_eigensystem](#)
- Procedure, public [inv](#) => [mqc_matrix_inverse](#)
- Procedure, public [det](#) => [mqc_matrix_determinant](#)
- Procedure, public [trace](#) => [mqc_matrix_trace](#)
- Procedure, public [rmsmax](#) => [mqc_matrix_rms_max](#)
- Procedure, public [sqrt](#) => [mqc_matrix_sqrt](#)
- Procedure, public [at](#) => [mqc_matrix_scalar_at](#)
- Procedure, public [vat](#) => [mqc_matrix_vector_at](#)
- Procedure, public [mat](#) => [mqc_matrix_matrix_at](#)
- Procedure, public [put](#) => [mqc_matrix_scalar_put](#)
- Procedure, public [vput](#) => [mqc_matrix_vector_put](#)
- Procedure, public [mput](#) => [mqc_matrix_matrix_put](#)
- Procedure, public [s_type](#) => [mqc_matrix_storagetype](#)

Public Attributes

- [real](#)(kind=real64), dimension(:,:), allocatable [matr](#)
- [integer](#)(kind=int64), dimension(:,:), allocatable [mati](#)
- [complex](#)(kind=real64), dimension(:,:), allocatable [matc](#)

6.28.1 Member Function/Subroutine Documentation

6.28.1.1 at()

Procedure, public mqc_algebra::mqc_matrix::at ()

6.28.1.2 dagger()

Procedure, public mqc_algebra::mqc_matrix::dagger ()

6.28.1.3 det()

Procedure, public mqc_algebra::mqc_matrix::det ()

6.28.1.4 diag()

Procedure, public mqc_algebra::mqc_matrix::diag ()

6.28.1.5 eigensys()

Procedure, public mqc_algebra::mqc_matrix::eigensys ()

6.28.1.6 identity()

Procedure, public mqc_algebra::mqc_matrix::identity ()

6.28.1.7 init()

Procedure, public mqc_algebra::mqc_matrix::init ()

6.28.1.8 initialize()

Procedure, public mqc_algebra::mqc_matrix::initialize ()

6.28.1.9 inv()

Procedure, public mqc_algebra::mqc_matrix::inv ()

6.28.1.10 mat()

Procedure, public mqc_algebra::mqc_matrix::mat ()

6.28.1.11 mput()

Procedure, public mqc_algebra::mqc_matrix::mput ()

6.28.1.12 norm()

Procedure, public mqc_algebra::mqc_matrix::norm ()

6.28.1.13 print()

Procedure, public mqc_algebra::mqc_matrix::print ()

6.28.1.14 put()

Procedure, public mqc_algebra::mqc_matrix::put ()

6.28.1.15 rmsmax()

Procedure, public mqc_algebra::mqc_matrix::rmsmax ()

6.28.1.16 s_type()

Procedure, public mqc_algebra::mqc_matrix::s_type ()

6.28.1.17 set()

Procedure, public mqc_algebra::mqc_matrix::set ()

6.28.1.18 sqrt()

Procedure, public mqc_algebra::mqc_matrix::sqrt ()

6.28.1.19 svd()

Procedure, public mqc_algebra::mqc_matrix::svd ()

6.28.1.20 trace()

Procedure, public mqc_algebra::mqc_matrix::trace ()

6.28.1.21 transpose()

Procedure, public mqc_algebra::mqc_matrix::transpose ()

6.28.1.22 vat()

Procedure, public mqc_algebra::mqc_matrix::vat ()

6.28.1.23 vput()

Procedure, public `mqc_algebra::mqc_matrix::vput ()`

6.28.2 Member Data Documentation

6.28.2.1 matc

`complex(kind=real64), dimension(:,,:), allocatable mqc_algebra::mqc_matrix::matc`

6.28.2.2 mati

`integer(kind=int64), dimension(:,,:), allocatable mqc_algebra::mqc_matrix::mati`

6.28.2.3 matr

`real(kind=real64), dimension(:,,:), allocatable mqc_algebra::mqc_matrix::matr`

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

- [src/mqc_algebra.F03](#)

6.29 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference

Public Member Functions

- subroutine [mqc_matrix_diagmatrix_put_integer](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_real](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_complex](#) (mat, diagMatrixIn)
- subroutine [mqc_matrix_diagmatrix_put_vector](#) (diagVectorIn, mat)

6.29.1 Member Function/Subroutine Documentation

6.29.1.1 mqc_matrix_diagmatrix_put_complex()

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

6.29.1.2 mqc_matrix_diagmatrix_put_integer()

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

6.29.1.3 mqc_matrix_diagmatrix_put_real()

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

6.29.1.4 mqc_matrix_diagmatrix_put_vector()

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

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

- [src/mqc_algebra.F03](#)

6.30 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference

Public Member Functions

- subroutine [mqc_matrix_symmmatrix_put_integer](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_real](#) (mat, symmMatrixIn)
- subroutine [mqc_matrix_symmmatrix_put_complex](#) (mat, symmMatrixIn)

6.30.1 Member Function/Subroutine Documentation

6.30.1.1 `mqc_matrix_symmmatrix_put_complex()`

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

6.30.1.2 `mqc_matrix_symmmatrix_put_integer()`

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

6.30.1.3 `mqc_matrix_symmmatrix_put_real()`

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

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

- [src/mqc_algebra.F03](#)

6.31 `mqc_est::mqc_matrix_undospinblockghf` Interface Reference

Public Member Functions

- subroutine [mqc_matrix_undospinblockghf_eigenvalues](#) (eigenvaluesIn, vectorOut)
- subroutine [mqc_matrix_undospinblockghf_integral](#) (integralln, matrixOut)

6.31.1 Member Function/Subroutine Documentation

6.31.1.1 mqc_matrix_undospinblockghf_eigenvalues()

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

6.31.1.2 mqc_matrix_undospinblockghf_integral()

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

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

- src/[mqc_est.F03](#)

6.32 mqc_algebra::mqc_print Interface Reference

Public Member Functions

- subroutine [mqc_print_scalar_algebra1](#) (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)
MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar
- subroutine [mqc_print_vector_algebra1](#) (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_print_matrix_algebra1](#) (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine [mqc_print_r4tensor_algebra1](#) (Tensor, IOut, Header, blank_at_top, blank_at_bottom)

6.32.1 Member Function/Subroutine Documentation

6.32.1.1 mqc_print_matrix_algebra1()

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

6.32.1.2 `mqc_print_r4tensor_algebra1()`

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

6.32.1.3 `mqc_print_scalar_algebra1()`

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.32.1.4 mqc_print_vector_algebra1()

```

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

```

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

- src/[mqc_algebra.F03](#)

6.33 mqc_est::mqc_print Interface Reference

Public Member Functions

- subroutine [mqc_print_wavefunction](#) (wavefunction, iOut, label)
- subroutine [mqc_print_integral](#) (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_eigenvalues](#) (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_print_twoeris](#) (twoERIs, iOut, header, blank_at_top, blank_at_bottom)

6.33.1 Member Function/Subroutine Documentation

6.33.1.1 mqc_print_eigenvalues()

```

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

```

6.33.1.2 mqc_print_integral()

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

6.33.1.3 mqc_print_twoeris()

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

6.33.1.4 mqc_print_wavefunction()

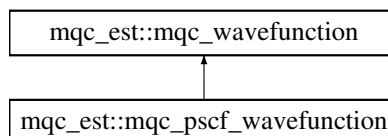
```
subroutine mqc_est::mqc_print::mqc_print_wavefunction (
    class(mqc_wavefunction) wavefunction,
    integer(kind=int64), intent(in) iOut,
    character(len=*), intent(in), optional label )
```

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

- [src/mqc_est.F03](#)

6.34 mqc_est::mqc_pscf_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_pscf_wavefunction:



Public Attributes

- integer(kind=int64) [ncore](#)
- integer(kind=int64) [nval](#)
- integer(kind=int64) [nactive](#)
- integer(kind=int64) [nfrz](#)
- type(mqc_matrix) [pscf_amplitudes](#)
- type(mqc_vector) [pscf_energies](#)

Additional Inherited Members

6.34.1 Member Data Documentation

6.34.1.1 nactive

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nactive
```

6.34.1.2 ncore

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::ncore
```

6.34.1.3 nfrz

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nfrz
```

6.34.1.4 nval

```
integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nval
```

6.34.1.5 pscf_amplitudes

```
type(mqc_matrix) mqc_est::mqc_pscf_wavefunction::pscf_amplitudes
```

6.34.1.6 pscf_energies

```
type(mqc_vector) mqc_est::mqc_pscf_wavefunction::pscf_energies
```

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

- src/[mqc_est.F03](#)

6.35 mqc_algebra::mqc_r4tensor Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_r4tensor_algebra1](#)
- Procedure, public [at](#) => [mqc_r4tensor_at](#)
- Procedure, public [put](#) => [mqc_r4tensor_put](#)
- Procedure, public [initialize](#) => [mqc_r4tensor_initialize](#)
- Procedure, public [init](#) => [mqc_r4tensor_initialize](#)

6.35.1 Member Function/Subroutine Documentation

6.35.1.1 at()

```
Procedure, public mqc_algebra::mqc_r4tensor::at ( )
```

6.35.1.2 init()

```
Procedure, public mqc_algebra::mqc_r4tensor::init ( )
```

6.35.1.3 initialize()

```
Procedure, public mqc_algebra::mqc_r4tensor::initialize ( )
```

6.35.1.4 print()

Procedure, public mqc_algebra::mqc_r4tensor::print ()

6.35.1.5 put()

Procedure, public mqc_algebra::mqc_r4tensor::put ()

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

- src/[mqc_algebra.F03](#)

6.36 mqc_algebra::mqc_scalar Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_scalar_algebra1](#)
- Procedure, public [rval](#) => [mqc_scalar_get_intrinsic_real](#)
- Procedure, public [ival](#) => [mqc_scalar_get_intrinsic_integer](#)
- Procedure, public [cval](#) => [mqc_scalar_get_intrinsic_complex](#)
- Procedure, public [abs](#) => [mqc_scalar_get_abs_value](#)
- Procedure, public [random](#) => [mqc_scalar_get_random_value](#)

6.36.1 Member Function/Subroutine Documentation

6.36.1.1 abs()

Procedure, public mqc_algebra::mqc_scalar::abs ()

6.36.1.2 cval()

Procedure, public mqc_algebra::mqc_scalar::cval ()

6.36.1.3 ival()

Procedure, public mqc_algebra::mqc_scalar::ival ()

6.36.1.4 print()

Procedure, public mqc_algebra::mqc_scalar::print ()

6.36.1.5 random()

Procedure, public mqc_algebra::mqc_scalar::random ()

6.36.1.6 rval()

Procedure, public mqc_algebra::mqc_scalar::rval ()

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

- [src/mqc_algebra.F03](#)

6.37 mqc_est::mqc_scf_eigenvalues Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_eigenvalues](#)
- Procedure, public [getlabel](#) => [mqc_eigenvalues_array_name](#)
- Procedure, public [addlabel](#) => [mqc_eigenvalues_add_name](#)
- Procedure, public [getblock](#) => [mqc_eigenvalues_output_block](#)
- Procedure, public [power](#) => [mqc_scf_eigenvalues_power](#)
- Procedure, public [at](#) => [mqc_eigenvalues_at](#)

6.37.1 Member Function/Subroutine Documentation

6.37.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::addlabel ( )
```

6.37.1.2 at()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::at ( )
```

6.37.1.3 getblock()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::getblock ( )
```

6.37.1.4 getlabel()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::getlabel ( )
```

6.37.1.5 power()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::power ( )
```

6.37.1.6 print()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::print ( )
```

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

- [src/mqc_est.F03](#)

6.38 mqc_est::mqc_scf_integral Type Reference

Public Member Functions

- Procedure, public [print](#) => [mqc_print_integral](#)
- Procedure, public [getlabel](#) => [mqc_integral_array_name](#)
- Procedure, public [addlabel](#) => [mqc_integral_add_name](#)
- Procedure, public [getblock](#) => [mqc_integral_output_block](#)
- Procedure, public [identity](#) => [mqc_integral_identity](#)
- Procedure, public [init](#) => [mqc_integral_initialize](#)
- Procedure, public [diag](#) => [mqc_scf_integral_diagonalize](#)
- Procedure, public [eigensys](#) => [mqc_scf_integral_generalized_eigensystem](#)
- Procedure, public [inv](#) => [mqc_scf_integral_inverse](#)
- Procedure, public [trace](#) => [mqc_scf_integral_trace](#)
- Procedure, public [det](#) => [mqc_scf_integral_determinant](#)
- Procedure, public [norm](#) => [mqc_integral_norm](#)
- Procedure, public [setelist](#) => [mqc_integral_set_energy_list](#)
- Procedure, public [getelist](#) => [mqc_integral_get_energy_list](#)
- Procedure, public [deleteelist](#) => [mqc_integral_delete_energy_list](#)
- Procedure, public [orbitals](#) => [mqc_integral_output_orbitals](#)
- Procedure, public [swap](#) => [mqc_integral_swap_orbitals](#)

6.38.1 Member Function/Subroutine Documentation

6.38.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_integral::addlabel ( )
```

6.38.1.2 deleteelist()

```
Procedure, public mqc_est::mqc_scf_integral::deleteelist ( )
```

6.38.1.3 det()

```
Procedure, public mqc_est::mqc_scf_integral::det ( )
```


6.38.1.4 diag()

Procedure, public mqc_est::mqc_scf_integral::diag ()

6.38.1.5 eigensys()

Procedure, public mqc_est::mqc_scf_integral::eigensys ()

6.38.1.6 getblock()

Procedure, public mqc_est::mqc_scf_integral::getblock ()

6.38.1.7 getelist()

Procedure, public mqc_est::mqc_scf_integral::getelist ()

6.38.1.8 getlabel()

Procedure, public mqc_est::mqc_scf_integral::getlabel ()

6.38.1.9 identity()

Procedure, public mqc_est::mqc_scf_integral::identity ()

6.38.1.10 init()

Procedure, public mqc_est::mqc_scf_integral::init ()

6.38.1.11 inv()

```
Procedure, public mqc_est::mqc_scf_integral::inv ( )
```

6.38.1.12 norm()

```
Procedure, public mqc_est::mqc_scf_integral::norm ( )
```

6.38.1.13 orbitals()

```
Procedure, public mqc_est::mqc_scf_integral::orbitals ( )
```

6.38.1.14 print()

```
Procedure, public mqc_est::mqc_scf_integral::print ( )
```

6.38.1.15 setelist()

```
Procedure, public mqc_est::mqc_scf_integral::setelist ( )
```

6.38.1.16 swap()

```
Procedure, public mqc_est::mqc_scf_integral::swap ( )
```

6.38.1.17 trace()

```
Procedure, public mqc_est::mqc_scf_integral::trace ( )
```

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

- [src/mqc_est.F03](#)

6.39 mqc_algebra::mqc_set_array2vector Interface Reference

Public Member Functions

- subroutine [mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)

6.39.1 Member Function/Subroutine Documentation

6.39.1.1 mqc_set_array2vector_complex()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_complex (
    type(mqc\_vector), intent(inout) VectorOut,
    complex(kind=real64), dimension(:), intent(in) ArrayIn )
```

6.39.1.2 mqc_set_array2vector_integer()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_integer (
    type(mqc\_vector), intent(inout) VectorOut,
    integer(kind=int64), dimension(:), intent(in) ArrayIn )
```

6.39.1.3 mqc_set_array2vector_real()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_real (
    type(mqc\_vector), intent(inout) VectorOut,
    real(kind=real64), dimension(:), intent(in) ArrayIn )
```

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

- src/[mqc_algebra.F03](#)

6.40 mqc_est::mqc_twoeris Type Reference

Public Member Functions

- procedure, public [print](#) => [mqc_print_twoeris](#)

6.40.1 Member Function/Subroutine Documentation

6.40.1.1 `print()`

```
procedure, public mqc_est::mqc_twoeris::print ( )
```

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

- [src/mqc_est.F03](#)

6.41 `mqc_algebra::mqc_vector` Type Reference

Public Member Functions

- Procedure, public `print` => `mqc_print_vector_algebra1`
- Procedure, public `initialize` => `mqc_vector_initialize`
- Procedure, public `size` => `mqc_length_vector`
- Procedure, public `init` => `mqc_vector_initialize`
- Procedure, public `norm` => `mqc_vector_norm`
- Procedure, public `transpose` => `mqc_vector_transpose`
- Procedure, public `dagger` => `mqc_vector_conjugate_transpose`
- Procedure, public `at` => `mqc_vector_scalar_at`
- Procedure, public `vat` => `mqc_vector_vector_at`
- Procedure, public `put` => `mqc_vector_scalar_put`
- Procedure, public `vput` => `mqc_vector_vector_put`
- Procedure, public `push` => `mqc_vector_push`
- Procedure, public `unshift` => `mqc_vector_unshift`
- Procedure, public `pop` => `mqc_vector_pop`
- Procedure, public `shift` => `mqc_vector_shift`
- Procedure, public `maxval` => `mqc_vector_maxval`
- Procedure, public `minval` => `mqc_vector_minloc`
- Procedure, public `maxloc` => `mqc_vector_maxval`
- Procedure, public `minloc` => `mqc_vector_minloc`
- Procedure, public `argsort` => `mqc_vector_argsort`
- Procedure, public `sort` => `mqc_vector_sort`
- Procedure, public `sqrt` => `mqc_vector_sqrt`
- Procedure, public `abs` => `mqc_vector_abs`
- Procedure, public `power` => `mqc_vector_power`
- Procedure, public `diag` => `mqc_matrix_diagmatrix_put_vector`

Public Attributes

- integer(kind=int64) [length](#) =0
- character(len=64) [data_type](#)
- [real](#)(kind=real64), dimension(:), allocatable [vecr](#)
- integer(kind=int64), dimension(:), allocatable [veci](#)
- complex(kind=real64), dimension(:), allocatable [vecc](#)

6.41.1 Member Function/Subroutine Documentation

6.41.1.1 `abs()`

Procedure, public mqc_algebra::mqc_vector::abs ()

6.41.1.2 `argsort()`

Procedure, public mqc_algebra::mqc_vector::argsort ()

6.41.1.3 `at()`

Procedure, public mqc_algebra::mqc_vector::at ()

6.41.1.4 `dagger()`

Procedure, public mqc_algebra::mqc_vector::dagger ()

6.41.1.5 `diag()`

Procedure, public mqc_algebra::mqc_vector::diag ()

6.41.1.6 init()

Procedure, public mqc_algebra::mqc_vector::init ()

6.41.1.7 initialize()

Procedure, public mqc_algebra::mqc_vector::initialize ()

6.41.1.8 maxloc()

Procedure, public mqc_algebra::mqc_vector::maxloc ()

6.41.1.9 maxval()

Procedure, public mqc_algebra::mqc_vector::maxval ()

6.41.1.10 minloc()

Procedure, public mqc_algebra::mqc_vector::minloc ()

6.41.1.11 minval()

Procedure, public mqc_algebra::mqc_vector::minval ()

6.41.1.12 norm()

Procedure, public mqc_algebra::mqc_vector::norm ()

6.41.1.13 pop()

Procedure, public mqc_algebra::mqc_vector::pop ()

6.41.1.14 power()

Procedure, public mqc_algebra::mqc_vector::power ()

6.41.1.15 print()

Procedure, public mqc_algebra::mqc_vector::print ()

6.41.1.16 push()

Procedure, public mqc_algebra::mqc_vector::push ()

6.41.1.17 put()

Procedure, public mqc_algebra::mqc_vector::put ()

6.41.1.18 shift()

Procedure, public mqc_algebra::mqc_vector::shift ()

6.41.1.19 size()

Procedure, public mqc_algebra::mqc_vector::size ()

6.41.1.20 sort()

```
Procedure, public mqc_algebra::mqc_vector::sort ( )
```

6.41.1.21 sqrt()

```
Procedure, public mqc_algebra::mqc_vector::sqrt ( )
```

6.41.1.22 transpose()

```
Procedure, public mqc_algebra::mqc_vector::transpose ( )
```

6.41.1.23 unshift()

```
Procedure, public mqc_algebra::mqc_vector::unshift ( )
```

6.41.1.24 vat()

```
Procedure, public mqc_algebra::mqc_vector::vat ( )
```

6.41.1.25 vput()

```
Procedure, public mqc_algebra::mqc_vector::vput ( )
```

6.41.2 Member Data Documentation**6.41.2.1 data_type**

```
character(len=64) mqc_algebra::mqc_vector::data_type
```


6.41.2.2 length

```
integer(kind=int64) mqc_algebra::mqc_vector::length =0
```

6.41.2.3 vecc

```
complex(kind=real64), dimension(:), allocatable mqc_algebra::mqc_vector::vecc
```

6.41.2.4 veci

```
integer(kind=int64), dimension(:), allocatable mqc_algebra::mqc_vector::veci
```

6.41.2.5 vecr

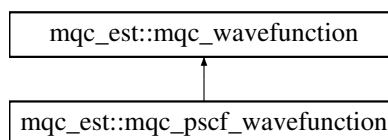
```
real(kind=real64), dimension(:), allocatable mqc_algebra::mqc_vector::vecr
```

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

- [src/mqc_algebra.F03](#)

6.42 mqc_est::mqc_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_wavefunction:



Public Member Functions

- Procedure, public [print](#) => [mqc_print_wavefunction](#)

Public Attributes

- type(mqc_scf_integral) `mo_coefficients`
- type(mqc_scf_eigenvalues) `mo_energies`
- type(mqc_scf_eigenvalues) `mo_symmetries`
- type(mqc_scf_integral) `core_hamiltonian`
- type(mqc_scf_integral) `fock_matrix`
- type(mqc_scf_integral) `density_matrix`
- type(mqc_scf_integral) `scf_density_matrix`
- type(mqc_scf_integral) `overlap_matrix`
- type(mqc_scalar) `nalpha`
- type(mqc_scalar) `nbeta`
- type(mqc_scalar) `nelectrons`
- type(mqc_scalar) `nbasis`
- type(mqc_scalar) `charge`
- type(mqc_scalar) `multiplicity`
- character(len=256) `basis`
- character(len=256) `symmetry`
- character(len=256) `wf_type`
- logical `wf_complex`

6.42.1 Member Function/Subroutine Documentation

6.42.1.1 `print()`

Procedure, public mqc_est::mqc_wavefunction::print ()

6.42.2 Member Data Documentation

6.42.2.1 `basis`

character(len=256) mqc_est::mqc_wavefunction::basis

6.42.2.2 `charge`

type(mqc_scalar) mqc_est::mqc_wavefunction::charge

6.42.2.3 core_hamiltonian

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::core_hamiltonian
```

6.42.2.4 density_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::density_matrix
```

6.42.2.5 fock_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::fock_matrix
```

6.42.2.6 mo_coefficients

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::mo_coefficients
```

6.42.2.7 mo_energies

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_energies
```

6.42.2.8 mo_symmetries

```
type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_symmetries
```

6.42.2.9 multiplicity

```
type(mqc_scalar) mqc_est::mqc_wavefunction::multiplicity
```

6.42.2.10 nalpha

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nalpha
```

6.42.2.11 nbasis

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbasis
```

6.42.2.12 nbeta

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nbeta
```

6.42.2.13 nelectrons

```
type(mqc_scalar) mqc_est::mqc_wavefunction::nelectrons
```

6.42.2.14 overlap_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::overlap_matrix
```

6.42.2.15 scf_density_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::scf_density_matrix
```

6.42.2.16 symmetry

```
character(len=256) mqc_est::mqc_wavefunction::symmetry
```

6.42.2.17 wf_complex

```
logical mqc_est::mqc_wavefunction::wf_complex
```

6.42.2.18 wf_type

```
character(len=256) mqc_est::mqc_wavefunction::wf_type
```

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

- [src/mqc_est.F03](#)

6.43 mqc_algebra::operator(*) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalarmultiply](#) (Scalar1, Scalar2)
MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
- type([mqc_scalar](#)) function [mqc_realscalarmultiply](#) (RealIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarrealmultiply](#) (Scalar, RealIn)
- type([mqc_scalar](#)) function [mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
- type([mqc_vector](#)) function [mqc_scalarvectorproduct](#) (Scalar, Vector)
- type([mqc_vector](#)) function [mqc_vectorscalarproduct](#) (vector, scalar)
- type([mqc_matrix](#)) function [mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type([mqc_matrix](#)) function [mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type([mqc_vector](#)) function [mqc_realvectorproduct](#) (RealIn, Vector)
- type([mqc_vector](#)) function [mqc_vectorrealproduct](#) (vector, realIn)
- type([mqc_vector](#)) function [mqc_integervectorproduct](#) (intIn, Vector)
- type([mqc_vector](#)) function [mqc_vectorintegerproduct](#) (vector, intIn)
- type([mqc_vector](#)) function [mqc_complexvectorproduct](#) (ComplexIn, Vector)
- type([mqc_vector](#)) function [mqc_vectorcomplexproduct](#) (vector, complexIn)
- type([mqc_matrix](#)) function [mqc_matrixmatrixproduct](#) (MA, MB)

6.43.1 Member Function/Subroutine Documentation

6.43.1.1 mqc_complexscalarmultiply()

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

6.43.1.2 mqc_complexvectorproduct()

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

6.43.1.3 mqc_integerscalarmultiply()

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

6.43.1.4 mqc_integervectorproduct()

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

6.43.1.5 mqc_matrixmatrixproduct()

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

6.43.1.6 mqc_matrixscalarproduct()

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

6.43.1.7 mqc_realscalarmultiply()

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

6.43.1.8 mqc_realvectorproduct()

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

6.43.1.9 mqc_scalarcomplexmultiply()

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

6.43.1.10 mqc_scalarintegermultiply()

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

6.43.1.11 mqc_scalarmatrixproduct()

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

6.43.1.12 mqc_scalarmultiply()

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

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

Purpose:

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

6.43.1.13 mqc_scalarrealmultiply()

```

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

```

6.43.1.14 mqc_scalarvectorproduct()

```

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

```

6.43.1.15 mqc_vectorcomplexproduct()

```

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

```


6.43.1.16 mqc_vectorintegerproduct()

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

6.43.1.17 mqc_vectorrealproduct()

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

6.43.1.18 mqc_vectorscalarproduct()

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

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

- [src/mqc_algebra.F03](#)

6.44 mqc_est::operator(*) Interface Reference**Public Member Functions**

- type(mqc_scf_integral) function [mqc_scalar_integral_multiply](#) (scalar, integral)
- type(mqc_scf_integral) function [mqc_integral_scalar_multiply](#) (integral, scalar)

6.44.1 Member Function/Subroutine Documentation**6.44.1.1 mqc_integral_scalar_multiply()**

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

6.44.1.2 `mqc_scalar_integral_multiply()`

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

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

- [src/mqc_est.F03](#)

6.45 `mqc_algebra::operator(**)` Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalarexponent](#) (Scalar1, Scalar2)
MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar
- type([mqc_scalar](#)) function [mqc_scalarintegerexponent](#) (Scalar, IntIn)
MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an intrinsic integer
- type([mqc_scalar](#)) function [mqc_scalarrealexponent](#) (Scalar, RealIn)
MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real
- type([mqc_scalar](#)) function [mqc_scalarcomplexexponent](#) (Scalar, Compln)
MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex

6.45.1 Member Function/Subroutine Documentation

6.45.1.1 `mqc_scalarcomplexexponent()`

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

6.45.1.2 mqc_scalarexponent()

```

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

```

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

6.45.1.3 mqc_scalarintegerexponent()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

6.45.1.4 mqc_scalarrealexponent()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.46 mqc_est::operator(+) Interface Reference

Public Member Functions

- [type\(mqc_scf_integral\)](#) function [mqc_integral_sum](#) (integralA, integralB)

6.46.1 Member Function/Subroutine Documentation

6.46.1.1 mqc_integral_sum()

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

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

- [src/mqc_est.F03](#)

6.47 mqc_algebra::operator(+) Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalaradd](#) (Scalar1, Scalar2)
MQC_ScalarAdd is a function that sums two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalaradd](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegeradd](#) (Scalar, IntegerIn)
- type([mqc_scalar](#)) function [mqc_realscalaradd](#) (RealIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarrealadd](#) (Scalar, RealIn)
- type([mqc_scalar](#)) function [mqc_complexscalaradd](#) (ComplexIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarcomplexadd](#) (Scalar, ComplexIn)
- type([mqc_vector](#)) function [mqc_vectorvectorsum](#) (Vector1In, Vector2In)
- type([mqc_vector](#)) function [mqc_scalarvectorsum](#) (ScalarIn, VectorIn)
- type([mqc_matrix](#)) function [mqc_matrixmatrixsum](#) (MA, MB)

6.47.1 Member Function/Subroutine Documentation

6.47.1.1 mqc_complexscalaradd()

```
type(mqc\_scalar) function mqc_algebra::operator(+)::mqc_complexscalaradd (
    complex(kind=real64), intent(in) ComplexIn,
    type(mqc\_scalar), intent(in) Scalar )
```

6.47.1.2 mqc_integerscalaradd()

```
type(mqc\_scalar) function mqc_algebra::operator(+)::mqc_integerscalaradd (
    integer(kind=int64), intent(in) IntegerIn,
    type(mqc\_scalar), intent(in) Scalar )
```

6.47.1.3 mqc_matrixmatrixsum()

```
type(mqc\_matrix) function mqc_algebra::operator(+)::mqc_matrixmatrixsum (
    type(mqc\_matrix), intent(in) MA,
    type(mqc\_matrix), intent(in) MB )
```

6.47.1.4 mqc_realscalaradd()

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

6.47.1.5 mqc_scalaradd()

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

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

Purpose:

MQC_ScalarAdd is a function that sums two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

6.47.1.6 mqc_scalarcomplexadd()

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

6.47.1.7 mqc_scalarintegeradd()

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

6.47.1.8 mqc_scalarrealadd()

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

6.47.1.9 mqc_scalarvectorsum()

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

6.47.1.10 mqc_vectorvectorsum()

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

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

- [src/mqc_algebra.F03](#)

6.48 mqc_est::operator(-) Interface Reference

Public Member Functions

- type(mqc_scf_integral) function [mqc_integral_difference](#) (integralA, integralB)

6.48.1 Member Function/Subroutine Documentation

6.48.1.1 mqc_integral_difference()

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

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

- src/mqc_est.F03

6.49 mqc_algebra::operator(-) Interface Reference

Public Member Functions

- type(mqc_scalar) function mqc_scalarsubtract (Scalar1, Scalar2)
MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects
- type(mqc_scalar) function mqc_integerscalarsubtract (IntegerIn, Scalar)
- type(mqc_scalar) function mqc_scalarintegersubtract (Scalar, IntegerIn)
- type(mqc_scalar) function mqc_realscalarsubtract (RealIn, Scalar)
- type(mqc_scalar) function mqc_scalarrealsubtract (Scalar, RealIn)
- type(mqc_scalar) function mqc_complexscalarsubtract (ComplexIn, Scalar)
- type(mqc_scalar) function mqc_scalarcomplexsubtract (Scalar, ComplexIn)
- type(mqc_vector) function mqc_vectorvectordifference (Vector1In, Vector2In)
- type(mqc_vector) function mqc_scalarvectordifference (ScalarIn, VectorIn)
- type(mqc_matrix) function mqc_matrixmatrixsubtract (MA, MB)

6.49.1 Member Function/Subroutine Documentation

6.49.1.1 mqc_complexscalarsubtract()

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

6.49.1.2 mqc_integerscalarsubtract()

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

6.49.1.3 mqc_matrixmatrixsubtract()

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

6.49.1.4 mqc_realscalarsubtract()

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

6.49.1.5 mqc_scalarcomplexsubtract()

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

6.49.1.6 mqc_scalarintegersubtract()

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

6.49.1.7 mqc_scalarrealsubtract()

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

6.49.1.8 mqc_scalarsubtract()

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

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

Purpose:

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

6.49.1.9 mqc_scalarvectordifference()

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

6.49.1.10 mqc_vectorvectordifference()

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

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

- [src/mqc_algebra.F03](#)

6.50 mqc_algebra::operator(.dot.) Interface Reference

Public Member Functions

- type(mqc_scalar) function [mqc_vectorvectordotproduct](#) (Vector1, Vector2)
- type(mqc_vector) function [mqc_vectormatrixdotproduct](#) (VA, MB)
- type(mqc_vector) function [mqc_matrixvectordotproduct](#) (MA, VB)
- type(mqc_matrix) function [mqc_matrixmatrixdotproduct](#) (MA, MB)

6.50.1 Member Function/Subroutine Documentation

6.50.1.1 `mqc_matrixmatrixdotproduct()`

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

6.50.1.2 `mqc_matrixvectordotproduct()`

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

6.50.1.3 `mqc_vectormatrixdotproduct()`

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

6.50.1.4 `mqc_vectorvectordotproduct()`

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

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

- [src/mqc_algebra.F03](#)

6.51 `mqc_algebra::operator(.eq.)` Interface Reference

Public Member Functions

- logical function [mqc_scalareq](#) (Scalar1, Scalar2)

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

6.51.1 Member Function/Subroutine Documentation

6.51.1.1 mqc_scalareq()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

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

- src/[mqc_algebra.F03](#)

6.52 mqc_algebra::operator(.ewd.) Interface Reference

Public Member Functions

- type([mqc_matrix](#)) function [mqc_elementmatrixdivide](#) (A, B)

6.52.1 Member Function/Subroutine Documentation

6.52.1.1 mqc_elementmatrixdivide()

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

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

- [src/mqc_algebra.F03](#)

6.53 mqc_algebra::operator(.ewp.) Interface Reference

Public Member Functions

- type(mqc_vector) function [mqc_elementvectorproduct](#) (Vector1In, Vector2In)
- type(mqc_matrix) function [mqc_elementmatrixproduct](#) (A, B)

6.53.1 Member Function/Subroutine Documentation

6.53.1.1 mqc_elementmatrixproduct()

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

6.53.1.2 mqc_elementvectorproduct()

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

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

- [src/mqc_algebra.F03](#)

6.54 `mqc_algebra::operator(.ge.)` Interface Reference

Public Member Functions

- logical function [mqc_scalarge](#) (Scalar1, Scalar2)

6.54.1 Member Function/Subroutine Documentation

6.54.1.1 `mqc_scalarge()`

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

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

- [src/mqc_algebra.F03](#)

6.55 `mqc_algebra::operator(.gt.)` Interface Reference

Public Member Functions

- logical function [mqc_scalargt](#) (Scalar1, Scalar2)
MQC_ScalarGT** is a function that returns **TRUE** if the left **MQC_Scalar** is greater than the right **MQC_Scalar
- logical function [mqc_scalargtinteger](#) (Scalar, Intln)
***MQC_ScalarGTInteger** is a function that returns **TRUE** if a **MQC_Scalar** is greater than an intrinsic integer*
- logical function [mqc_integertscalar](#) (Intln, Scalar)
MQC_IntegerGTScalar** is a function that returns **TRUE** if an intrinsic integer is greater than a **MQC_Scalar
- logical function [mqc_scalargtreal](#) (Scalar, Realln)
- logical function [mqc_realgtscalar](#) (Realln, Scalar)

6.55.1 Member Function/Subroutine Documentation

6.55.1.1 `mqc_integertgtscalar()`

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the intrinsic integer is greater than the real part of the MQC_Scalar and FALSE if the intrinsic integer is less than the real part of the MQC_Scalar. If the intrinsic integer is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is less than zero and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

6.55.1.2 `mqc_realgtgtscalar()`

```
logical function mqc_algebra::operator(.gt.)::mqc_realgtgtscalar (
    real(kind=real64), intent(in) RealIn,
    type(mqc_scalar), intent(in) Scalar )
```


6.55.1.3 mqc_scalargt()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the left real part is greater than the right real part and FALSE if the left real part is less than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is greater than the right imaginary part and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2016

6.55.1.4 mqc_scalargtinteger()

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

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

Purpose:

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

When dealing with complex numbers, the function returns `TRUE` if the real part of the `MQC_Scalar` is greater than the intrinsic integer and `FALSE` if the real part of the `MQC_Scalar` is less than the intrinsic integer. If the real part of the `MQC_Scalar` is equal to the intrinsic integer, the function returns `TRUE` if the imaginary part of `MQC_Scalar` is greater than zero and `FALSE` otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

6.55.1.5 `mqc_scalargtreal()`

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

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

- [src/mqc_algebra.F03](#)

6.56 `mqc_algebra::operator(.le.)` Interface Reference**Public Member Functions**

- logical function [mqc_scalarle](#) (`Scalar1`, `Scalar2`)
- logical function [mqc_scalarlereal](#) (`Scalar`, `RealIn`)
- logical function [mqc_reallscalar](#) (`RealIn`, `Scalar`)
- logical function [mqc_scalarleinteger](#) (`Scalar`, `IntIn`)
- logical function [mqc_integerlescalar](#) (`IntIn`, `Scalar`)

6.56.1 Member Function/Subroutine Documentation

6.56.1.1 mqc_integerlescalar()

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

6.56.1.2 mqc_reallescalar()

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

6.56.1.3 mqc_scalarle()

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

6.56.1.4 mqc_scalarleinteger()

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

6.56.1.5 mqc_scalarlereal()

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

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

- [src/mqc_algebra.F03](#)

6.57 mqc_algebra::operator(.lt.) Interface Reference

Public Member Functions

- logical function [mqc_scalarlt](#) (Scalar1, Scalar2)
MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar
- logical function [mqc_scalarltreal](#) (Scalar, RealIn)
MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real
- logical function [mqc_realltscalar](#) (RealIn, Scalar)
MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

6.57.1 Member Function/Subroutine Documentation

6.57.1.1 mqc_realltscalar()

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the intrinsic real is less than the real part of the MQC_Scalar and FALSE if the intrinsic real is greater than the real part of the MQC_Scalar. If the intrinsic real is equal to the real part of the MQC_Scalar, the function returns TRUE if the imaginary part of MQC_Scalar is greater than zero and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

6.57.1.2 `mqc_scalarlt()`

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

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

Purpose:

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

When dealing with complex numbers, the function returns TRUE if the left real part is less than the right real part and FALSE if the left real part is greater than the right real part. If the left real part is equal to the right real part, the function returns TRUE if the left imaginary part is less than the right imaginary part and FALSE otherwise. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2016

6.57.1.3 mqc_scalarltreal()

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

MQC_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. Note that this is the same procedure used in python.

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/[mqc_algebra.F03](#)

6.58 mqc_algebra::operator(.ne.) Interface Reference

Public Member Functions

- logical function [mqc_scalarne](#) (Scalar1, Scalar2)

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

6.58.1 Member Function/Subroutine Documentation

6.58.1.1 `mqc_scalarne()`

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

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

- [src/mqc_algebra.F03](#)

6.59 `mqc_algebra::operator(.outer.)` Interface Reference

Public Member Functions

- `type(mqc_matrix)` function `mqc_outer` (VA, VB)

6.59.1 Member Function/Subroutine Documentation

6.59.1.1 `mqc_outer()`

```
type(mqc\_matrix) function mqc_algebra::operator(.outer.):mqc_outer (
    type(mqc\_vector), intent(in) VA,
    type(mqc\_vector), intent(in) VB )
```

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

- [src/mqc_algebra.F03](#)

6.60 `mqc_algebra::operator(.x.)` Interface Reference

Public Member Functions

- type([mqc_vector](#)) function [mqc_crossproduct](#) (Vector1In, Vector2In)

6.60.1 Member Function/Subroutine Documentation

6.60.1.1 `mqc_crossproduct()`

```
type(mqc\_vector) function mqc_algebra::operator(.x.):mqc_crossproduct (
    type(mqc\_vector), intent(in) Vector1In,
    type(mqc\_vector), intent(in) Vector2In )
```

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

- [src/mqc_algebra.F03](#)

6.61 `mqc_algebra::operator(/)` Interface Reference

Public Member Functions

- type([mqc_scalar](#)) function [mqc_scalardivide](#) (Scalar1, Scalar2)
MQC_ScalarDivide is a function that divides two MQC_Scalar objects
- type([mqc_scalar](#)) function [mqc_integerscalardivide](#) (IntegerIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarintegerdivide](#) (Scalar, IntegerIn)
- type([mqc_scalar](#)) function [mqc_realscalardivide](#) (RealIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarrealdivide](#) (Scalar, RealIn)
- type([mqc_scalar](#)) function [mqc_complexscalardivide](#) (ComplexIn, Scalar)
- type([mqc_scalar](#)) function [mqc_scalarcomplexdivide](#) (Scalar, ComplexIn)
- type([mqc_vector](#)) function [mqc_vectorscalardivide](#) (vector, scalar)
- type([mqc_vector](#)) function [mqc_vectorrealdivide](#) (vector, realIn)
- type([mqc_vector](#)) function [mqc_vectorintegerdivide](#) (vector, intIn)
- type([mqc_vector](#)) function [mqc_vectorcomplexdivide](#) (vector, compln)

6.61.1 Member Function/Subroutine Documentation

6.61.1.1 mqc_complexscalardivide()

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

6.61.1.2 mqc_integerscalardivide()

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

6.61.1.3 mqc_realscalardivide()

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

6.61.1.4 mqc_scalarcomplexdivide()

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

6.61.1.5 mqc_scalardivide()

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

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

Purpose:

MQC_ScalarDivide is a function that divides MQC_Scalar objects.

Parameters

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

Author

L. M. Thompson

Date

2016

6.61.1.6 mqc_scalarintegerdivide()

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

6.61.1.7 mqc_scalarrealddivide()

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

6.61.1.8 mqc_vectorcomplexdivide()

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

6.61.1.9 mqc_vectorintegerdivide()

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

6.61.1.10 mqc_vectorrealddivide()

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

6.61.1.11 mqc_vectorscalardivide()

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

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

- [src/mqc_algebra.F03](#)

6.62 mqc_algebra::real Interface Reference

Public Member Functions

- `type(mqc_scalar) function mqc_scalar_complex_realpart (ScalarIn)`
- `type(mqc_vector) function mqc_vector_complex_realpart (A)`

6.62.1 Member Function/Subroutine Documentation

6.62.1.1 mqc_scalar_complex_realpart()

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

6.62.1.2 mqc_vector_complex_realpart()

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

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

- src/mqc_algebra.F03

6.63 mqc_algebra::sin Interface Reference

Public Member Functions

- type(mqc_scalar) function mqc_scalar_sin (Scalar)
MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

6.63.1 Member Function/Subroutine Documentation

6.63.1.1 mqc_scalar_sin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- [src/mqc_algebra.F03](#)

6.64 mqc_algebra::sqrt Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_sqrt` (Scalar)
MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

6.64.1 Member Function/Subroutine Documentation

6.64.1.1 mqc_scalar_sqrt()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

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

- [src/mqc_algebra.F03](#)

6.65 mqc_algebra::tan Interface Reference

Public Member Functions

- `type(mqc_scalar)` function `mqc_scalar_tan` (Scalar)
MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar

6.65.1 Member Function/Subroutine Documentation

6.65.1.1 mqc_scalar_tan()

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

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

Purpose:

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

Parameters

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](#)

6.66 `mqc_est::transpose` Interface Reference

Public Member Functions

- `type(mqc_scf_integral)` function `mqc_integral_transpose` (`integral`, `label`)

6.66.1 Member Function/Subroutine Documentation

6.66.1.1 `mqc_integral_transpose()`

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

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

- [src/mqc_est.F03](#)

6.67 `mqc_algebra::transpose` Interface Reference

Public Member Functions

- `type(mqc_vector)` function `mqc_vector_transpose` (`Vector`)
- `type(mqc_matrix)` function `mqc_matrix_transpose` (`Matrix`)

6.67.1 Member Function/Subroutine Documentation

6.67.1.1 `mqc_matrix_transpose()`

```
type(mqc\_matrix) function mqc_algebra::transpose::mqc_matrix_transpose (  
    class(mqc\_matrix), intent(in) Matrix )
```

6.67.1.2 `mqc_vector_transpose()`

```
type(mqc\_vector) function mqc_algebra::transpose::mqc_vector_transpose (  
    class(mqc\_vector), intent(in) Vector )
```

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

- [src/mqc_algebra.F03](#)

Chapter 7

File Documentation

7.1 src/mqc_algebra.F03 File Reference

Data Types

- type [mqc_algebra::mqc_scalar](#)
- type [mqc_algebra::mqc_vector](#)
- type [mqc_algebra::mqc_matrix](#)
- type [mqc_algebra::mqc_r4tensor](#)
- interface [mqc_algebra::mqc_print](#)
- interface [mqc_algebra::contraction](#)
- interface [mqc_algebra::conjg](#)
- interface [mqc_algebra::mqc_have_real](#)
- interface [mqc_algebra::mqc_have_int](#)
- interface [mqc_algebra::mqc_have_complex](#)
- interface [mqc_algebra::mqc_cast_real](#)
- interface [mqc_algebra::mqc_cast_complex](#)
- interface [mqc_algebra::matmul](#)
- interface [mqc_algebra::transpose](#)
- interface [mqc_algebra::dagger](#)
- interface [mqc_algebra::cmplx](#)
- interface [mqc_algebra::sqrt](#)
- interface [mqc_algebra::abs](#)
- interface [mqc_algebra::real](#)
- interface [mqc_algebra::aimag](#)
- interface [mqc_algebra::sin](#)
- interface [mqc_algebra::cos](#)
- interface [mqc_algebra::tan](#)
- interface [mqc_algebra::asin](#)
- interface [mqc_algebra::acos](#)
- interface [mqc_algebra::atan](#)
- interface [mqc_algebra::atan2](#)
- interface [mqc_algebra::mqc_set_array2vector](#)
- interface [mqc_algebra::mqc_matrix_symmmatrix_put](#)

- interface [mqc_algebra::mqc_matrix_diagmatrix_put](#)
- interface [mqc_algebra::matrix_symm2sq](#)
- interface [mqc_algebra::dot_product](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(/\)](#)
- interface [mqc_algebra::operator\(**\)](#)
- interface [mqc_algebra::operator\(.ne.\)](#)
- interface [mqc_algebra::operator\(.eq.\)](#)
- interface [mqc_algebra::operator\(.lt.\)](#)
- interface [mqc_algebra::operator\(.gt.\)](#)
- interface [mqc_algebra::operator\(.le.\)](#)
- interface [mqc_algebra::operator\(.ge.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(.dot.\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(/\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(.ewp.\)](#)
- interface [mqc_algebra::operator\(.ewd.\)](#)
- interface [mqc_algebra::operator\(.x.\)](#)
- interface [mqc_algebra::operator\(.outer.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)
- interface [mqc_algebra::operator\(+\)](#)
- interface [mqc_algebra::operator\(-\)](#)
- interface [mqc_algebra::operator\(*\)](#)
- interface [mqc_algebra::operator\(.dot.\)](#)
- interface [mqc_algebra::assignment\(=\)](#)

Modules

- module [mqc_algebra](#)

Functions/Subroutines

- integer(kind=int64) function [mqc_algebra::factorial](#) (n)
Factorial returns the factorial of an integer
- integer(kind=int64) function [mqc_algebra::bin_coeff](#) (N, K)
Bin_Coeff returns the binomial coefficient of (n,k)
- subroutine [mqc_algebra::mqc_allocate_scalar](#) (Scalar, Data_type)
MQC_Allocate_Scalar is used to allocate a scalar type variable of the MQC_Scalar class
- subroutine [mqc_algebra::mqc_deallocate_scalar](#) (Scalar)
MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class
- logical function [mqc_algebra::mqc_scalar_isallocated](#) (Scalar)
MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar

- subroutine [mqc_algebra::mqc_input_integer_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Integer_Scalar** is a subroutine is used to set an intrinsic integer to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_input_real_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Real_Scalar** is a subroutine is used to set an intrinsic real to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_input_complex_scalar](#) (ScalarOut, ScalarIn)
***MQC_Input_Complex_Scalar** is a subroutine is used to set an intrinsic complex to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_mqcscalar_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output MQCScalar_Scalar** is a subroutine used to output an MQC_scalar equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_integer_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Integer_Scalar** is a subroutine used to output an intrinsic integer equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_real_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Real_Scalar** is a subroutine used to output an intrinsic real equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_output_complex_scalar](#) (ScalarOut, ScalarIn)
***MQC_Output_Complex_Scalar** is a subroutine used to output an intrinsic complex equal to an MQC_Scalar*
- subroutine [mqc_algebra::mqc_print_scalar_algebra1](#) (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)
***MQC_Print_Scalar_Algebra1** is a subroutine used to print an MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_cmplx](#) (Scalar1, Scalar2)
***MQC_Scalar_Cmplx** is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_sqrt](#) (Scalar)
***MQC_Scalar_Sqrt** is a function used to return the square root of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_sin](#) (Scalar)
***MQC_Scalar_Sin** is a function used to return the sine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_cos](#) (Scalar)
***MQC_Scalar_Cos** is a function used to return the cosine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_tan](#) (Scalar)
***MQC_Scalar_Tan** is a function used to return the tangent of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_asin](#) (Scalar)
***MQC_Scalar_ASin** is a function used to return the arcsin of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_acos](#) (Scalar)
***MQC_Scalar_ACos** is a function used to return the arccosine of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_atan](#) (Scalar)
***MQC_Scalar_ATan** is a function used to return the arctangent of an MQC_scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_atan2](#) (Scalar)
***MQC_Scalar_ATan2** is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram*
- logical function [mqc_algebra::mqc_scalar_havereal](#) (Scalar)
***MQC_Scalar_HaveReal** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real*
- logical function [mqc_algebra::mqc_scalar_haveinteger](#) (Scalar)
***MQC_Scalar_HaveInteger** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer*
- logical function [mqc_algebra::mqc_scalar_havecomplex](#) (Scalar)
***MQC_Scalar_HaveComplex** is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex*
- real(kind=real64) function [mqc_algebra::mqc_scalar_get_intrinsic_real](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Real** is a function that returns the MQC_scalar value as an intrinsic real*
- integer(kind=int64) function [mqc_algebra::mqc_scalar_get_intrinsic_integer](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Integer** is a function that returns the MQC_scalar value as an intrinsic integer*

- complex(kind=real64) function [mqc_algebra::mqc_scalar_get_intrinsic_complex](#) (Scalar)
***MQC_Scalar_Get_Intrinsic_Complex** is a function that returns the MQC_scalar value as an intrinsic complex*
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_get_abs_value](#) (Scalar)
***MQC_Scalar_Get_ABS_Value** is a function that returns the absolute value of MQC_scalar variable*
- subroutine [mqc_algebra::mqc_scalar_get_random_value](#) (Scalar)
***MQC_Scalar_Get_Random_Value** is a function that returns a random real value from a uniform distribution between zero and one*
- type(mqc_scalar) function [mqc_algebra::mqc_scalaradd](#) (Scalar1, Scalar2)
***MQC_ScalarAdd** is a function that sums two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarsubtract](#) (Scalar1, Scalar2)
***MQC_ScalarSubtract** is a function that subtracts two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarmultiply](#) (Scalar1, Scalar2)
***MQC_ScalarMultiply** is a function that multiplies two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalardivide](#) (Scalar1, Scalar2)
***MQC_ScalarDivide** is a function that divides two MQC_Scalar objects*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarexponent](#) (Scalar1, Scalar2)
***MQC_ScalarExponent** is a function that raises one MQC_Scalar to the power of another MQC_Scalar*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegerexponent](#) (Scalar, IntIn)
***MQC_ScalarIntegerExponent** is a function that raises an MQC_Scalar to the power of an intrinsic integer*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealexponent](#) (Scalar, RealIn)
***MQC_ScalarRealExponent** is a function that raises an MQC_Scalar to the power of an intrinsic real*
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexexponent](#) (Scalar, ComplIn)
***MQC_ScalarComplexExponent** is a function that raises an MQC_Scalar to the power of an intrinsic complex*
- logical function [mqc_algebra::mqc_scalarne](#) (Scalar1, Scalar2)
***MQC_ScalarNE** is a function that returns TRUE if two MQC_Scalar variables are not equal*
- logical function [mqc_algebra::mqc_scalareq](#) (Scalar1, Scalar2)
***MQC_ScalarEQ** is a function that returns TRUE if two MQC_Scalar variables are equal*
- logical function [mqc_algebra::mqc_scalarlt](#) (Scalar1, Scalar2)
***MQC_ScalarLT** is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar*
- logical function [mqc_algebra::mqc_realltscalar](#) (RealIn, Scalar)
***MQC_RealLTScalar** is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar*
- logical function [mqc_algebra::mqc_scalarltreal](#) (Scalar, RealIn)
***MQC_ScalarLTReal** is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real*
- logical function [mqc_algebra::mqc_scalargt](#) (Scalar1, Scalar2)
***MQC_ScalarGT** is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar*
- logical function [mqc_algebra::mqc_integergtscalar](#) (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)
- logical function [mqc_algebra::mqc_scalargtreal](#) (Scalar, RealIn)
- logical function [mqc_algebra::mqc_scalarle](#) (Scalar1, Scalar2)
- logical function [mqc_algebra::mqc_reallescalar](#) (RealIn, Scalar)
- logical function [mqc_algebra::mqc_scalarlereal](#) (Scalar, RealIn)
- logical function [mqc_algebra::mqc_integerlescalar](#) (IntIn, Scalar)
- logical function [mqc_algebra::mqc_scalarleinteger](#) (Scalar, IntIn)
- logical function [mqc_algebra::mqc_scalarge](#) (Scalar1, Scalar2)
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_conjugate](#) (ScalarIn)

- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_realpart](#) (ScalarIn)
- type(mqc_scalar) function [mqc_algebra::mqc_scalar_complex_imagpart](#) (ScalarIn)
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalarmultiply](#) (IntegerIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegermultiply](#) (Scalar, IntegerIn)
- type(mqc_scalar) function [mqc_algebra::mqc_realscalarmultiply](#) (RealIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealmultiply](#) (Scalar, RealIn)
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalarmultiply](#) (ComplexIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexmultiply](#) (Scalar, ComplexIn)
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalardivide](#) (IntegerIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegerdivide](#) (Scalar, IntegerIn)
- type(mqc_scalar) function [mqc_algebra::mqc_realscalardivide](#) (RealIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealddivide](#) (Scalar, RealIn)
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalardivide](#) (ComplexIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexdivide](#) (Scalar, ComplexIn)
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalaradd](#) (IntegerIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegeradd](#) (Scalar, IntegerIn)
- type(mqc_scalar) function [mqc_algebra::mqc_realscalaradd](#) (RealIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealadd](#) (Scalar, RealIn)
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalaradd](#) (ComplexIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexadd](#) (Scalar, ComplexIn)
- type(mqc_scalar) function [mqc_algebra::mqc_integerscalarsubtract](#) (IntegerIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarintegersubtract](#) (Scalar, IntegerIn)
- type(mqc_scalar) function [mqc_algebra::mqc_realscalarsubtract](#) (RealIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarrealsubtract](#) (Scalar, RealIn)
- type(mqc_scalar) function [mqc_algebra::mqc_complexscalarsubtract](#) (ComplexIn, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_scalarcomplexsubtract](#) (Scalar, ComplexIn)
- subroutine [mqc_algebra::mqc_allocate_vector](#) (N, Vector, Data_Type)
- subroutine [mqc_algebra::mqc_deallocate_vector](#) (Vector)
- integer(kind=int64) function [mqc_algebra::mqc_length_vector](#) (Vector)
- logical function [mqc_algebra::mqc_vector_havereal](#) (Vector)
- logical function [mqc_algebra::mqc_vector_haveinteger](#) (Vector)
- logical function [mqc_algebra::mqc_vector_havecomplex](#) (Vector)
- logical function [mqc_algebra::mqc_vector_iscolumn](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_int2real](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_int2complex](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_real2int](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_real2complex](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_complex2int](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_copy_complex2real](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_scalar_at](#) (Vec, I)
- type(mqc_vector) function [mqc_algebra::mqc_vector_vector_at](#) (Vec, I, J)
- subroutine [mqc_algebra::mqc_set_vector2integerarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_vector2realarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_vector2complexarray](#) (ArrayOut, VectorIn)
- subroutine [mqc_algebra::mqc_set_array2vector_integer](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_array2vector_real](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_array2vector_complex](#) (VectorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_set_vector2vector](#) (VectorOut, VectorIn)
- type(mqc_vector) function [mqc_algebra::mqc_vectorvectorsum](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_vectorvectordifference](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_scalarvectorsum](#) (ScalarIn, VectorIn)

- type(mqc_vector) function [mqc_algebra::mqc_scalarvectordifference](#) (ScalarIn, VectorIn)
- type(mqc_vector) function [mqc_algebra::mqc_elementvectorproduct](#) (Vector1In, Vector2In)
- type(mqc_vector) function [mqc_algebra::mqc_vector_transpose](#) (Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vector_conjugate_transpose](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vectorvectordotproduct](#) (Vector1, Vector2)
- type(mqc_matrix) function [mqc_algebra::mqc_outer](#) (VA, VB)
- type(mqc_vector) function [mqc_algebra::mqc_crossproduct](#) (Vector1In, Vector2In)
- subroutine [mqc_algebra::mqc_print_vector_algebra1](#) (Vector, IOOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
- type(mqc_vector) function [mqc_algebra::mqc_vector_cast_real](#) (VA)
- type(mqc_vector) function [mqc_algebra::mqc_vector_cast_complex](#) (VA)
- subroutine [mqc_algebra::mqc_vector_scalar_put](#) (Vector, Scalar, I)
- subroutine [mqc_algebra::mqc_vector_scalar_increment](#) (Vector, Scalar, I)
- subroutine [mqc_algebra::mqc_vector_vector_put](#) (Vector, VectorIn, I)
- subroutine [mqc_algebra::mqc_vector_initialize](#) (Vector, Length, Scalar)
- type(mqc_vector) function [mqc_algebra::mqc_scalarvectorproduct](#) (Scalar, Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vectorscalarproduct](#) (vector, scalar)
- type(mqc_vector) function [mqc_algebra::mqc_vectorscalardivide](#) (vector, scalar)
- type(mqc_vector) function [mqc_algebra::mqc_realvectorproduct](#) (Realln, Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vectorrealproduct](#) (vector, realln)
- type(mqc_vector) function [mqc_algebra::mqc_vectorrealdivide](#) (vector, realln)
- type(mqc_vector) function [mqc_algebra::mqc_integervectorproduct](#) (intln, Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vectorintegerproduct](#) (vector, intln)
- type(mqc_vector) function [mqc_algebra::mqc_vectorintegerdivide](#) (vector, intln)
- type(mqc_vector) function [mqc_algebra::mqc_complexvectorproduct](#) (Compln, Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vectorcomplexproduct](#) (vector, compln)
- type(mqc_vector) function [mqc_algebra::mqc_vectorcomplexdivide](#) (vector, compln)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_norm](#) (vector, methodIn)
- logical function [mqc_algebra::mqc_vector_isallocated](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_push](#) (Vector, Scalar)
- subroutine [mqc_algebra::mqc_vector_unshift](#) (Vector, Scalar)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_pop](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_shift](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_maxval](#) (Vector)
- type(mqc_scalar) function [mqc_algebra::mqc_vector_minval](#) (Vector)
- integer function [mqc_algebra::mqc_vector_maxloc](#) (Vector)
- integer function [mqc_algebra::mqc_vector_minloc](#) (Vector)
- type(mqc_vector) function [mqc_algebra::mqc_vector_argsort](#) (Vector)
- subroutine [mqc_algebra::mqc_vector_sort](#) (Vector, idx)
- subroutine [mqc_algebra::mqc_vector_sqrt](#) (A)
- type(mqc_vector) function [mqc_algebra::mqc_vector_abs](#) (A)
- subroutine [mqc_algebra::mqc_vector_power](#) (A, P)
- type(mqc_vector) function [mqc_algebra::mqc_vector_complex_realpart](#) (A)
- type(mqc_vector) function [mqc_algebra::mqc_vector_complex_imagpart](#) (A)
- type(mqc_vector) function [mqc_algebra::mqc_vector_cmplx](#) (Vector1, Vector2)
- character(len=64) function [mqc_algebra::mqc_matrix_storage_type](#) (Matrix)
- subroutine [mqc_algebra::mqc_matrix_diagonalize](#) (A, EVals, EVecs)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_cast_real](#) (MA)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_cast_complex](#) (MA)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_scalar_at](#) (Mat, I, J)
- type(mqc_vector) function [mqc_algebra::mqc_matrix_vector_at](#) (Mat, Rows, Cols)

- recursive subroutine `mqc_algebra::mqc_matrix_vector_put` (Mat, VectorIn, Rows, Cols)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_matrix_at` (Mat, Rows, Cols)
- MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix***
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_vector` (diagVectorIn, mat)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_integer` (mat, diagMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_real` (mat, diagMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_diagmatrix_put_complex` (mat, diagMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_symmmatrix_put_integer` (mat, symmMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_symmmatrix_put_real` (mat, symmMatrixIn)
- subroutine `mqc_algebra::mqc_matrix_symmmatrix_put_complex` (mat, symmMatrixIn)
- recursive subroutine `mqc_algebra::mqc_matrix_matrix_put` (Mat, MatrixIn, Rows, Cols)
- integer(kind=int64) function `mqc_algebra::symindexhash` (i, j, k, l)
- type(mqc_matrix) function `mqc_algebra::mqc_elementmatrixproduct` (A, B)
- type(mqc_matrix) function `mqc_algebra::mqc_elementmatrixdivide` (A, B)
- logical function `mqc_algebra::mqc_matrix_test_symmetric` (Matrix, Option)
- logical function `mqc_algebra::mqc_matrix_test_diagonal` (Matrix)
- subroutine `mqc_algebra::mqc_allocate_matrix` (M, N, Matrix, Data_Type, Storage)
- subroutine `mqc_algebra::mqc_deallocate_matrix` (Matrix)
- logical function `mqc_algebra::mqc_matrix_isallocated` (Matrix)
- subroutine `mqc_algebra::mqc_set_integerarray2matrix` (MatrixOut, ArrayIn)
- subroutine `mqc_algebra::mqc_set_realarray2matrix` (MatrixOut, ArrayIn)
- subroutine `mqc_algebra::mqc_set_complexarray2matrix` (MatrixOut, ArrayIn)
- subroutine `mqc_algebra::mqc_set_matrix2integerarray` (ArrayOut, MatrixIn)
- subroutine `mqc_algebra::mqc_set_matrix2realarray` (ArrayOut, MatrixIn)
- subroutine `mqc_algebra::mqc_set_matrix2complexarray` (ArrayOut, MatrixIn)
- subroutine `mqc_algebra::mqc_set_matrix2matrix` (MatrixOut, MatrixIn)
- subroutine `mqc_algebra::mqc_print_matrix_algebra1` (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
- subroutine `mqc_algebra::mqc_matrix_copy_int2real` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_copy_int2complex` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_copy_real2int` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_copy_real2complex` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_copy_complex2int` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_copy_complex2real` (Matrix)
- integer(kind=int64) function `mqc_algebra::mqc_matrix_rows` (Matrix)
- integer(kind=int64) function `mqc_algebra::mqc_matrix_columns` (Matrix)
- logical function `mqc_algebra::mqc_matrix_havereal` (Matrix)
- logical function `mqc_algebra::mqc_matrix_haveinteger` (Matrix)
- logical function `mqc_algebra::mqc_matrix_havecomplex` (Matrix)
- logical function `mqc_algebra::mqc_matrix_havfull` (Matrix)
- logical function `mqc_algebra::mqc_matrix_havesymmetric` (Matrix)
- logical function `mqc_algebra::mqc_matrix_havediagonal` (Matrix)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_transpose` (Matrix)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_conjugate_transpose` (Matrix)
- type(mqc_matrix) function `mqc_algebra::mqc_matrix_symmetrize` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_full2symm` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_symm2full` (Matrix, Option)
- subroutine `mqc_algebra::mqc_matrix_full2diag` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_diag2full` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_symm2diag` (Matrix)
- subroutine `mqc_algebra::mqc_matrix_diag2symm` (Matrix)

- type(mqc_matrix) function [mqc_algebra::mqc_matrix_symm2full_func](#) (Matrix)
- subroutine [mqc_algebra::matrix_symm2sq_integer](#) (N, I_Symm, I_Sq)
- subroutine [mqc_algebra::matrix_symm2sq_real](#) (N, A_Symm, A_Sq)
- subroutine [mqc_algebra::matrix_symm2sq_complex](#) (N, A_Symm, A_Sq)
- type(mqc_matrix) function [mqc_algebra::mqc_vector2diagmatrix](#) (vector)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixsum](#) (MA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixsubtract](#) (MA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixproduct](#) (MA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixmatrixdotproduct](#) (MA, MB)
- type(mqc_vector) function [mqc_algebra::mqc_matrixvectordotproduct](#) (MA, VB)
- type(mqc_vector) function [mqc_algebra::mqc_vectormatrixdotproduct](#) (VA, MB)
- type(mqc_matrix) function [mqc_algebra::mqc_matrixscalarproduct](#) (Matrix, Scalar)
- type(mqc_matrix) function [mqc_algebra::mqc_scalarmatrixproduct](#) (Scalar, Matrix)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_matrix_contraction](#) (Matrix1, Matrix2)
- subroutine [mqc_algebra::mqc_matrix_scalar_put](#) (Matrix, Scalar, I, J)
- subroutine [mqc_algebra::mqc_matrix_initialize](#) (Matrix, Rows, Columns, Scalar, Storage)
- subroutine [mqc_algebra::mqc_matrix_identity](#) (matrix, n, m)
- subroutine [mqc_algebra::mqc_matrix_set](#) (matrix, scalar, storage)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_norm](#) (matrix, methodIn)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_determinant](#) (a)
- type(mqc_matrix) function [mqc_algebra::mqc_matrix_inverse](#) (a)
- type(mqc_scalar) function [mqc_algebra::mqc_matrix_trace](#) (matrix)
- subroutine [mqc_algebra::mqc_matrix_generalized_eigensystem](#) (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine [mqc_algebra::mqc_matrix_svd](#) (A, EVals, EUVecs, EVVecs)
- subroutine [mqc_algebra::mqc_matrix_rms_max](#) (A, rms_A, max_A)
- subroutine [mqc_algebra::mqc_matrix_sqrt](#) (A, eVals, eVecs)
- type(mqc_matrix) function [mqc_algebra::mqc_givens_matrix](#) (m_size, angle, p, q)
- subroutine [mqc_algebra::mqc_allocate_r4tensor](#) (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine [mqc_algebra::mqc_deallocate_r4tensor](#) (Tensor)
- type(mqc_scalar) function [mqc_algebra::mqc_r4tensor_at](#) (Tensor, I, J, K, L)
- subroutine [mqc_algebra::mqc_r4tensor_put](#) (Tensor, Element, I, J, K, L)
- subroutine [mqc_algebra::mqc_print_r4tensor_algebra1](#) (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine [mqc_algebra::mqc_set_array2tensor](#) (TensorOut, ArrayIn)
- subroutine [mqc_algebra::mqc_r4tensor_initialize](#) (R4Tensor, I, J, K, L, Scalar)
- subroutine [mqc_algebra::mqc_matrix_symmsymmr4tensor_put_real](#) (r4Tensor, symmSymmMatrixIn)
- subroutine [mqc_algebra::mqc_matrix_symmsymmr4tensor_put_complex](#) (r4Tensor, symmSymmMatrixIn)
- logical function [mqc_algebra::mqc_r4tensor_haveinteger](#) (R4Tensor)
- logical function [mqc_algebra::mqc_r4tensor_havereal](#) (R4Tensor)
- logical function [mqc_algebra::mqc_r4tensor_havecomplex](#) (R4Tensor)

7.2 src/mqc_est.F03 File Reference

Data Types

- type [mqc_est::mqc_scf_integral](#)
- type [mqc_est::mqc_scf_eigenvalues](#)
- type [mqc_est::mqc_wavefunction](#)
- type [mqc_est::mqc_pscf_wavefunction](#)

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

Modules

- module [mqc_est](#)

Functions/Subroutines

- subroutine [mqc_est::mqc_print_wavefunction](#) (wavefunction, iOut, label)
- subroutine [mqc_est::mqc_print_integral](#) (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_est::mqc_print_eigenvalues](#) (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine [mqc_est::mqc_print_twoeris](#) (twoERIs, iOut, header, blank_at_top, blank_at_bottom)
- logical function [mqc_est::mqc_integral_isallocated](#) (Integral)
- logical function [mqc_est::mqc_eigenvalues_isallocated](#) (Eigenvalues)
- logical function [mqc_est::mqc_integral_has_alpha](#) (integral)
- logical function [mqc_est::mqc_integral_has_beta](#) (integral)
- logical function [mqc_est::mqc_integral_has_alphabeta](#) (integral)
- logical function [mqc_est::mqc_integral_has_betaalpha](#) (integral)
- logical function [mqc_est::mqc_eigenvalues_has_alpha](#) (eigenvalues)
- logical function [mqc_est::mqc_eigenvalues_has_beta](#) (eigenvalues)
- character(len=64) function [mqc_est::mqc_integral_array_type](#) (integral)
- character(len=64) function [mqc_est::mqc_eigenvalues_array_type](#) (eigenvalues)
- character(len=64) function [mqc_est::mqc_integral_array_name](#) (integral)
- character(len=64) function [mqc_est::mqc_eigenvalues_array_name](#) (eigenvalues)
- subroutine [mqc_est::mqc_integral_add_name](#) (integral, arrayName)
- subroutine [mqc_est::mqc_eigenvalues_add_name](#) (eigenvalues, arrayName)
- integer(kind=int64) function [mqc_est::mqc_integral_dimension](#) (integral, label, axis)
- integer(kind=int64) function [mqc_est::mqc_eigenvalues_dimension](#) (eigenvalues, label)
- subroutine [mqc_est::mqc_twoeris_allocate](#) (twoERIs, storageType, integralType, alpha, beta, alphaBeta, beta \leftrightarrow 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, betaOrbsIn, 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, integralIn)
- 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, lEVecs)
- 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, [143](#)
 - mqc_algebra::mqc_vector, [151](#)
- addlabel
 - mqc_est::mqc_scf_eigenvalues, [144](#)
 - mqc_est::mqc_scf_integral, [146](#)
- alpha
 - mqc_est::mqc_determinant_string, [127](#)
- argsort
 - mqc_algebra::mqc_vector, [151](#)
- at
 - mqc_algebra::mqc_matrix, [130](#)
 - mqc_algebra::mqc_r4tensor, [142](#)
 - mqc_algebra::mqc_vector, [151](#)
 - mqc_est::mqc_scf_eigenvalues, [145](#)
- basis
 - mqc_est::mqc_wavefunction, [156](#)
- beta
 - mqc_est::mqc_determinant_string, [127](#)
- bin_coeff
 - mqc_algebra, [16](#)
- charge
 - mqc_est::mqc_wavefunction, [156](#)
- core_hamiltonian
 - mqc_est::mqc_wavefunction, [156](#)
- cval
 - mqc_algebra::mqc_scalar, [143](#)
- dagger
 - mqc_algebra::mqc_matrix, [130](#)
 - mqc_algebra::mqc_vector, [151](#)
- data_type
 - mqc_algebra::mqc_vector, [154](#)
- deleteelist
 - mqc_est::mqc_scf_integral, [146](#)
- density_matrix
 - mqc_est::mqc_wavefunction, [157](#)
- det
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_est::mqc_scf_integral, [146](#)
- diag
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_algebra::mqc_vector, [151](#)
 - mqc_est::mqc_scf_integral, [146](#)
- eigensys
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_est::mqc_scf_integral, [147](#)
- factorial
 - mqc_algebra, [17](#)
- fock_matrix
 - mqc_est::mqc_wavefunction, [157](#)
- gen_det_str
 - mqc_est, [86](#)
- get_one_gamma_matrix
 - mqc_est, [86](#)
- getblock
 - mqc_est::mqc_scf_eigenvalues, [145](#)
 - mqc_est::mqc_scf_integral, [147](#)
- getelist
 - mqc_est::mqc_scf_integral, [147](#)
- getlabel
 - mqc_est::mqc_scf_eigenvalues, [145](#)
 - mqc_est::mqc_scf_integral, [147](#)
- identity
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_est::mqc_scf_integral, [147](#)
- init
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_algebra::mqc_r4tensor, [142](#)
 - mqc_algebra::mqc_vector, [151](#)
 - mqc_est::mqc_scf_integral, [147](#)
- initialize
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_algebra::mqc_r4tensor, [142](#)
 - mqc_algebra::mqc_vector, [152](#)
- inv
 - mqc_algebra::mqc_matrix, [131](#)
 - mqc_est::mqc_scf_integral, [147](#)
- ival
 - mqc_algebra::mqc_scalar, [143](#)
- length
 - mqc_algebra::mqc_vector, [154](#)
- mat
 - mqc_algebra::mqc_matrix, [132](#)
- matc

- mqc_algebra::mqc_matrix, 134
- mati
 - mqc_algebra::mqc_matrix, 134
- matr
 - mqc_algebra::mqc_matrix, 134
- matrix_symm2sq_complex
 - mqc_algebra, 17
 - mqc_algebra::matrix_symm2sq, 124
- matrix_symm2sq_integer
 - mqc_algebra, 17
 - mqc_algebra::matrix_symm2sq, 124
- matrix_symm2sq_real
 - mqc_algebra, 18
 - mqc_algebra::matrix_symm2sq, 124
- maxloc
 - mqc_algebra::mqc_vector, 152
- maxval
 - mqc_algebra::mqc_vector, 152
- minloc
 - mqc_algebra::mqc_vector, 152
- minval
 - mqc_algebra::mqc_vector, 152
- mo_coefficients
 - mqc_est::mqc_wavefunction, 157
- mo_energies
 - mqc_est::mqc_wavefunction, 157
- mo_symmetries
 - mqc_est::mqc_wavefunction, 157
- mput
 - mqc_algebra::mqc_matrix, 132
- mqc_algebra, 9
 - bin_coeff, 16
 - factorial, 17
 - matrix_symm2sq_complex, 17
 - matrix_symm2sq_integer, 17
 - matrix_symm2sq_real, 18
 - mqc_allocate_matrix, 18
 - mqc_allocate_r4tensor, 18
 - mqc_allocate_scalar, 18
 - mqc_allocate_vector, 19
 - mqc_complexscalaradd, 19
 - mqc_complexscalardivide, 20
 - mqc_complexscalarmultiply, 20
 - mqc_complexscalarsubtract, 20
 - mqc_complexvectorproduct, 20
 - mqc_crossproduct, 20
 - mqc_deallocate_matrix, 20
 - mqc_deallocate_r4tensor, 21
 - mqc_deallocate_scalar, 21
 - mqc_deallocate_vector, 21
 - mqc_elementmatrixdivide, 21
 - mqc_elementmatrixproduct, 22
 - mqc_elementvectorproduct, 22
 - mqc_givens_matrix, 22
 - mqc_input_complex_scalar, 22
 - mqc_input_integer_scalar, 23
 - mqc_input_real_scalar, 24
 - mqc_integertscalar, 24
 - mqc_integerlescalar, 25
 - mqc_integerscalaradd, 25
 - mqc_integerscalardivide, 25
 - mqc_integerscalarmultiply, 26
 - mqc_integerscalarsubtract, 26
 - mqc_integervectorproduct, 26
 - mqc_length_vector, 26
 - mqc_matrix_cast_complex, 26
 - mqc_matrix_cast_real, 26
 - mqc_matrix_columns, 27
 - mqc_matrix_conjugate_transpose, 27
 - mqc_matrix_copy_complex2int, 27
 - mqc_matrix_copy_complex2real, 27
 - mqc_matrix_copy_int2complex, 27
 - mqc_matrix_copy_int2real, 27
 - mqc_matrix_copy_real2complex, 28
 - mqc_matrix_copy_real2int, 28
 - mqc_matrix_determinant, 28
 - mqc_matrix_diag2full, 28
 - mqc_matrix_diag2symm, 28
 - mqc_matrix_diagmatrix_put_complex, 28
 - mqc_matrix_diagmatrix_put_integer, 29
 - mqc_matrix_diagmatrix_put_real, 29
 - mqc_matrix_diagmatrix_put_vector, 29
 - mqc_matrix_diagonalize, 29
 - mqc_matrix_full2diag, 29
 - mqc_matrix_full2symm, 29
 - mqc_matrix_generalized_eigensystem, 30
 - mqc_matrix_havecomplex, 30
 - mqc_matrix_havediagonal, 30
 - mqc_matrix_havetfull, 30
 - mqc_matrix_haveinteger, 30
 - mqc_matrix_havereal, 30
 - mqc_matrix_havesymmetric, 31
 - mqc_matrix_identity, 31
 - mqc_matrix_initialize, 31
 - mqc_matrix_inverse, 31
 - mqc_matrix_isallocated, 31
 - mqc_matrix_matrix_at, 31
 - mqc_matrix_matrix_contraction, 33
 - mqc_matrix_matrix_put, 33
 - mqc_matrix_norm, 33
 - mqc_matrix_rms_max, 34
 - mqc_matrix_rows, 34
 - mqc_matrix_scalar_at, 34
 - mqc_matrix_scalar_put, 34
 - mqc_matrix_set, 34
 - mqc_matrix_sqrt, 35
 - mqc_matrix_storagetype, 35
 - mqc_matrix_svd, 35

`mqc_matrix_symm2diag`, 35
`mqc_matrix_symm2full`, 35
`mqc_matrix_symm2full_func`, 36
`mqc_matrix_symmetrize`, 36
`mqc_matrix_symmmatrix_put_complex`, 36
`mqc_matrix_symmmatrix_put_integer`, 36
`mqc_matrix_symmmatrix_put_real`, 36
`mqc_matrix_symmsymmr4tensor_put_complex`, 36
`mqc_matrix_symmsymmr4tensor_put_real`, 37
`mqc_matrix_test_diagonal`, 37
`mqc_matrix_test_symmetric`, 37
`mqc_matrix_trace`, 37
`mqc_matrix_transpose`, 37
`mqc_matrix_vector_at`, 37
`mqc_matrix_vector_put`, 38
`mqc_matrixmatrixdotproduct`, 38
`mqc_matrixmatrixproduct`, 38
`mqc_matrixmatrixsubtract`, 38
`mqc_matrixmatrixsum`, 38
`mqc_matrixscalarproduct`, 39
`mqc_matrixvectordotproduct`, 39
`mqc_outer`, 39
`mqc_output_complex_scalar`, 39
`mqc_output_integer_scalar`, 40
`mqc_output_mqcscalar_scalar`, 41
`mqc_output_real_scalar`, 41
`mqc_print_matrix_algebra1`, 42
`mqc_print_r4tensor_algebra1`, 42
`mqc_print_scalar_algebra1`, 42
`mqc_print_vector_algebra1`, 43
`mqc_r4tensor_at`, 44
`mqc_r4tensor_havecomplex`, 44
`mqc_r4tensor_haveinteger`, 44
`mqc_r4tensor_havereal`, 44
`mqc_r4tensor_initialize`, 44
`mqc_r4tensor_put`, 45
`mqc_realgtscalar`, 45
`mqc_realltscalar`, 45
`mqc_realscalaradd`, 46
`mqc_realscalardivide`, 46
`mqc_realscalarmultiply`, 47
`mqc_realscalarsubtract`, 47
`mqc_realvectorproduct`, 47
`mqc_scalar_acos`, 47
`mqc_scalar_asin`, 48
`mqc_scalar_atan`, 48
`mqc_scalar_atan2`, 49
`mqc_scalar_cmplx`, 50
`mqc_scalar_complex_conjugate`, 50
`mqc_scalar_complex_imagpart`, 50
`mqc_scalar_complex_realpart`, 51
`mqc_scalar_cos`, 51
`mqc_scalar_get_abs_value`, 51
`mqc_scalar_get_intrinsic_complex`, 52
`mqc_scalar_get_intrinsic_integer`, 53
`mqc_scalar_get_intrinsic_real`, 53
`mqc_scalar_get_random_value`, 54
`mqc_scalar_havecomplex`, 54
`mqc_scalar_haveinteger`, 55
`mqc_scalar_havereal`, 56
`mqc_scalar_isallocated`, 56
`mqc_scalar_sin`, 57
`mqc_scalar_sqrt`, 58
`mqc_scalar_tan`, 58
`mqc_scalaradd`, 59
`mqc_scalarcomplexadd`, 59
`mqc_scalarcomplexdivide`, 60
`mqc_scalarcomplexexponent`, 60
`mqc_scalarcomplexmultiply`, 60
`mqc_scalarcomplexsubtract`, 61
`mqc_scalardivide`, 61
`mqc_scalareq`, 61
`mqc_scalarexponent`, 62
`mqc_scalarge`, 63
`mqc_scalargt`, 63
`mqc_scalargtinteger`, 64
`mqc_scalargtreal`, 65
`mqc_scalarintegeradd`, 65
`mqc_scalarintegerdivide`, 65
`mqc_scalarintegerexponent`, 65
`mqc_scalarintegermultiply`, 66
`mqc_scalarintegersubtract`, 66
`mqc_scalarle`, 66
`mqc_scalarleinteger`, 66
`mqc_scalarlereal`, 67
`mqc_scalarlt`, 67
`mqc_scalarltreal`, 68
`mqc_scalarmatrixproduct`, 68
`mqc_scalarmultiply`, 68
`mqc_scalarne`, 69
`mqc_scalarrealadd`, 70
`mqc_scalarrealddivide`, 70
`mqc_scalarrealexponent`, 70
`mqc_scalarrealmultiply`, 71
`mqc_scalarrealsubtract`, 71
`mqc_scalarsubtract`, 71
`mqc_scalarvectordifference`, 72
`mqc_scalarvectorproduct`, 72
`mqc_scalarvectorsum`, 72
`mqc_set_array2tensor`, 72
`mqc_set_array2vector_complex`, 73
`mqc_set_array2vector_integer`, 73
`mqc_set_array2vector_real`, 73
`mqc_set_complexarray2matrix`, 73
`mqc_set_integerarray2matrix`, 73
`mqc_set_matrix2complexarray`, 73
`mqc_set_matrix2integerarray`, 74

- mqc_set_matrix2matrix, 74
- mqc_set_matrix2realarray, 74
- mqc_set_realarray2matrix, 74
- mqc_set_vector2complexarray, 74
- mqc_set_vector2integerarray, 74
- mqc_set_vector2realarray, 75
- mqc_set_vector2vector, 75
- mqc_vector2diagmatrix, 75
- mqc_vector_abs, 75
- mqc_vector_argsort, 75
- mqc_vector_cast_complex, 75
- mqc_vector_cast_real, 76
- mqc_vector_cmplx, 76
- mqc_vector_complex_imagpart, 76
- mqc_vector_complex_realpart, 76
- mqc_vector_conjugate_transpose, 76
- mqc_vector_copy_complex2int, 76
- mqc_vector_copy_complex2real, 77
- mqc_vector_copy_int2complex, 77
- mqc_vector_copy_int2real, 77
- mqc_vector_copy_real2complex, 77
- mqc_vector_copy_real2int, 77
- mqc_vector_havecomplex, 77
- mqc_vector_haveinteger, 78
- mqc_vector_havereal, 78
- mqc_vector_initialize, 78
- mqc_vector_isallocated, 78
- mqc_vector_iscolumn, 78
- mqc_vector_maxloc, 78
- mqc_vector_maxval, 79
- mqc_vector_minloc, 79
- mqc_vector_minval, 79
- mqc_vector_norm, 79
- mqc_vector_pop, 79
- mqc_vector_power, 79
- mqc_vector_push, 80
- mqc_vector_scalar_at, 80
- mqc_vector_scalar_increment, 80
- mqc_vector_scalar_put, 80
- mqc_vector_shift, 80
- mqc_vector_sort, 80
- mqc_vector_sqrt, 81
- mqc_vector_transpose, 81
- mqc_vector_unshift, 81
- mqc_vector_vector_at, 81
- mqc_vector_vector_put, 81
- mqc_vectorcomplexdivide, 81
- mqc_vectorcomplexproduct, 82
- mqc_vectorintegerdivide, 82
- mqc_vectorintegerproduct, 82
- mqc_vectormatrixdotproduct, 82
- mqc_vectorrealdivide, 82
- mqc_vectorrealproduct, 82
- mqc_vectorscalardivide, 83
- mqc_vectorscalarproduct, 83
- mqc_vectorvectordifference, 83
- mqc_vectorvectordotproduct, 83
- mqc_vectorvectorsum, 83
- symindexhash, 83
- mqc_algebra::abs, 101
 - mqc_scalar_get_abs_value, 101
 - mqc_vector_abs, 102
- mqc_algebra::acos, 102
 - mqc_scalar_acos, 102
- mqc_algebra::aimag, 103
 - mqc_scalar_complex_imagpart, 103
 - mqc_vector_complex_imagpart, 103
- mqc_algebra::asin, 104
 - mqc_scalar_asin, 104
- mqc_algebra::assignment(=), 105
 - mqc_input_complex_scalar, 105
 - mqc_input_integer_scalar, 106
 - mqc_input_real_scalar, 107
 - mqc_output_complex_scalar, 107
 - mqc_output_integer_scalar, 108
 - mqc_output_mqcscalar_scalar, 109
 - mqc_output_real_scalar, 109
 - mqc_set_array2tensor, 110
 - mqc_set_array2vector_complex, 110
 - mqc_set_array2vector_integer, 110
 - mqc_set_array2vector_real, 110
 - mqc_set_complexarray2matrix, 111
 - mqc_set_integerarray2matrix, 111
 - mqc_set_matrix2complexarray, 111
 - mqc_set_matrix2integerarray, 111
 - mqc_set_matrix2matrix, 111
 - mqc_set_matrix2realarray, 111
 - mqc_set_realarray2matrix, 112
 - mqc_set_vector2complexarray, 112
 - mqc_set_vector2integerarray, 112
 - mqc_set_vector2realarray, 112
 - mqc_set_vector2vector, 112
- mqc_algebra::atan, 113
 - mqc_scalar_atan, 113
- mqc_algebra::atan2, 114
 - mqc_scalar_atan2, 114
- mqc_algebra::cmplx, 115
 - mqc_scalar_cmplx, 115
 - mqc_vector_cmplx, 116
- mqc_algebra::conjg, 116
 - mqc_scalar_complex_conjugate, 116
- mqc_algebra::contraction, 117
 - mqc_matrix_matrix_contraction, 117
- mqc_algebra::cos, 118
 - mqc_scalar_cos, 118
- mqc_algebra::dagger, 119
 - mqc_matrix_conjugate_transpose, 119
 - mqc_vector_conjugate_transpose, 119

- mqc_algebra::dot_product, 120
 - mqc_vectorvectordotproduct, 120
- mqc_algebra::matmul, 121
 - mqc_matrixmatrixdotproduct, 121
 - mqc_matrixvectordotproduct, 121
 - mqc_vectormatrixdotproduct, 122
- mqc_algebra::matrix_symm2sq, 124
 - matrix_symm2sq_complex, 124
 - matrix_symm2sq_integer, 124
 - matrix_symm2sq_real, 124
- mqc_algebra::mqc_cast_complex, 125
 - mqc_matrix_cast_complex, 125
 - mqc_vector_cast_complex, 125
- mqc_algebra::mqc_cast_real, 125
 - mqc_matrix_cast_real, 125
 - mqc_vector_cast_real, 126
- mqc_algebra::mqc_have_complex, 128
 - mqc_matrix_havecomplex, 128
 - mqc_vector_havecomplex, 128
- mqc_algebra::mqc_have_int, 128
 - mqc_matrix_haveinteger, 128
 - mqc_vector_haveinteger, 129
- mqc_algebra::mqc_have_real, 129
 - mqc_matrix_havereal, 129
 - mqc_vector_havereal, 129
- mqc_algebra::mqc_matrix, 130
 - at, 130
 - dagger, 130
 - det, 131
 - diag, 131
 - eigensys, 131
 - identity, 131
 - init, 131
 - initialize, 131
 - inv, 131
 - mat, 132
 - matc, 134
 - mati, 134
 - matr, 134
 - mput, 132
 - norm, 132
 - print, 132
 - put, 132
 - rmsmax, 132
 - s_type, 132
 - set, 133
 - sqrt, 133
 - svd, 133
 - trace, 133
 - transpose, 133
 - vat, 133
 - vput, 133
- mqc_algebra::mqc_matrix_diagmatrix_put, 134
 - mqc_matrix_diagmatrix_put_complex, 134
 - mqc_matrix_diagmatrix_put_integer, 135
 - mqc_matrix_diagmatrix_put_real, 135
 - mqc_matrix_diagmatrix_put_vector, 135
- mqc_algebra::mqc_matrix_symmmatrix_put, 135
 - mqc_matrix_symmmatrix_put_complex, 136
 - mqc_matrix_symmmatrix_put_integer, 136
 - mqc_matrix_symmmatrix_put_real, 136
- mqc_algebra::mqc_print, 137
 - mqc_print_matrix_algebra1, 137
 - mqc_print_r4tensor_algebra1, 137
 - mqc_print_scalar_algebra1, 138
 - mqc_print_vector_algebra1, 139
- mqc_algebra::mqc_r4tensor, 142
 - at, 142
 - init, 142
 - initialize, 142
 - print, 142
 - put, 143
- mqc_algebra::mqc_scalar, 143
 - abs, 143
 - cval, 143
 - ival, 143
 - print, 144
 - random, 144
 - rval, 144
- mqc_algebra::mqc_set_array2vector, 149
 - mqc_set_array2vector_complex, 149
 - mqc_set_array2vector_integer, 149
 - mqc_set_array2vector_real, 149
- mqc_algebra::mqc_vector, 150
 - abs, 151
 - argsort, 151
 - at, 151
 - dagger, 151
 - data_type, 154
 - diag, 151
 - init, 151
 - initialize, 152
 - length, 154
 - maxloc, 152
 - maxval, 152
 - minloc, 152
 - minval, 152
 - norm, 152
 - pop, 152
 - power, 153
 - print, 153
 - push, 153
 - put, 153
 - shift, 153
 - size, 153
 - sort, 153
 - sqrt, 154
 - transpose, 154

- unshift, 154
- vat, 154
- vecc, 155
- veci, 155
- vecr, 155
- vput, 154
- mqc_algebra::operator(**), 164
 - mqc_scalarcomplexexponent, 164
 - mqc_scalarexponent, 165
 - mqc_scalarintegerexponent, 166
 - mqc_scalarrealexponent, 166
- mqc_algebra::operator(*), 159
 - mqc_complexscalarmultiply, 159
 - mqc_complexvectorproduct, 160
 - mqc_integerscalarmultiply, 160
 - mqc_integervectorproduct, 160
 - mqc_matrixmatrixproduct, 160
 - mqc_matrixscalarproduct, 160
 - mqc_realscalarmultiply, 160
 - mqc_realvectorproduct, 161
 - mqc_scalarcomplexmultiply, 161
 - mqc_scalarintegermultiply, 161
 - mqc_scalarmatrixproduct, 161
 - mqc_scalarmultiply, 161
 - mqc_scalarrealmultiply, 162
 - mqc_scalarvectorproduct, 162
 - mqc_vectorcomplexproduct, 162
 - mqc_vectorintegerproduct, 162
 - mqc_vectorrealproduct, 163
 - mqc_vectorscalarproduct, 163
- mqc_algebra::operator(+), 168
 - mqc_complexscalaradd, 168
 - mqc_integerscalaradd, 168
 - mqc_matrixmatrixsum, 168
 - mqc_realscalaradd, 168
 - mqc_scalaradd, 169
 - mqc_scalarcomplexadd, 169
 - mqc_scalarintegeradd, 169
 - mqc_scalarrealadd, 170
 - mqc_scalarvectorsum, 170
 - mqc_vectorvectorsum, 170
- mqc_algebra::operator(-), 171
 - mqc_complexscalarsubtract, 171
 - mqc_integerscalarsubtract, 171
 - mqc_matrixmatrixsubtract, 171
 - mqc_realscalarsubtract, 172
 - mqc_scalarcomplexsubtract, 172
 - mqc_scalarintegersubtract, 172
 - mqc_scalarrealsubtract, 172
 - mqc_scalarsubtract, 172
 - mqc_scalarvectordifference, 173
 - mqc_vectorvectordifference, 173
- mqc_algebra::operator(.dot.), 173
 - mqc_matrixmatrixdotproduct, 174
 - mqc_matrixvectordotproduct, 174
 - mqc_vectormatrixdotproduct, 174
 - mqc_vectorvectordotproduct, 174
- mqc_algebra::operator(.eq.), 174
 - mqc_scalareq, 175
- mqc_algebra::operator(.ewd.), 175
 - mqc_elementmatrixdivide, 176
- mqc_algebra::operator(.ewp.), 176
 - mqc_elementmatrixproduct, 176
 - mqc_elementvectorproduct, 176
- mqc_algebra::operator(.ge.), 177
 - mqc_scalarge, 177
- mqc_algebra::operator(.gt.), 177
 - mqc_integergtscalar, 177
 - mqc_realgtscalar, 178
 - mqc_scalargt, 178
 - mqc_scalargtinteger, 179
 - mqc_scalargtreal, 180
- mqc_algebra::operator(.le.), 180
 - mqc_integerlescalar, 181
 - mqc_reallescalar, 181
 - mqc_scalarle, 181
 - mqc_scalarleinteger, 181
 - mqc_scalarlereal, 181
- mqc_algebra::operator(.lt.), 182
 - mqc_realltscalar, 182
 - mqc_scalarlt, 183
 - mqc_scalarltreal, 183
- mqc_algebra::operator(.ne.), 184
 - mqc_scalarne, 185
- mqc_algebra::operator(.outer.), 185
 - mqc_outer, 186
- mqc_algebra::operator(.x.), 186
 - mqc_crossproduct, 186
- mqc_algebra::operator(/), 186
 - mqc_complexscalardivide, 187
 - mqc_integerscalardivide, 187
 - mqc_realscalardivide, 187
 - mqc_scalarcomplexdivide, 187
 - mqc_scalardivide, 187
 - mqc_scalarintegerdivide, 188
 - mqc_scalarrealdivide, 188
 - mqc_vectorcomplexdivide, 188
 - mqc_vectorintegerdivide, 188
 - mqc_vectorrealdivide, 189
 - mqc_vectorscalardivide, 189
- mqc_algebra::real, 189
 - mqc_scalar_complex_realpart, 189
 - mqc_vector_complex_realpart, 189
- mqc_algebra::sin, 190
 - mqc_scalar_sin, 190
- mqc_algebra::sqrt, 191
 - mqc_scalar_sqrt, 191
- mqc_algebra::tan, 192

- mqc_scalar_tan, 192
- mqc_algebra::transpose, 193
 - mqc_matrix_transpose, 193
 - mqc_vector_transpose, 194
- mqc_allocate_matrix
 - mqc_algebra, 18
- mqc_allocate_r4tensor
 - mqc_algebra, 18
- mqc_allocate_scalar
 - mqc_algebra, 18
- mqc_allocate_vector
 - mqc_algebra, 19
- mqc_build_ci_hamiltonian
 - mqc_est, 86
- mqc_complexscalaradd
 - mqc_algebra, 19
 - mqc_algebra::operator(+), 168
- mqc_complexscalardivide
 - mqc_algebra, 20
 - mqc_algebra::operator(/), 187
- mqc_complexscalarmultiply
 - mqc_algebra, 20
 - mqc_algebra::operator(*), 159
- mqc_complexscalarsubtract
 - mqc_algebra, 20
 - mqc_algebra::operator(-), 171
- mqc_complexvectorproduct
 - mqc_algebra, 20
 - mqc_algebra::operator(*), 160
- mqc_crossproduct
 - mqc_algebra, 20
 - mqc_algebra::operator(.x.), 186
- mqc_deallocate_matrix
 - mqc_algebra, 20
- mqc_deallocate_r4tensor
 - mqc_algebra, 21
- mqc_deallocate_scalar
 - mqc_algebra, 21
- mqc_deallocate_vector
 - mqc_algebra, 21
- mqc_eigenvalue_eigenvalue_dotproduct
 - mqc_est, 86
 - mqc_est::dot_product, 121
- mqc_eigenvalues_add_name
 - mqc_est, 87
- mqc_eigenvalues_allocate
 - mqc_est, 87
- mqc_eigenvalues_array_name
 - mqc_est, 87
- mqc_eigenvalues_array_type
 - mqc_est, 87
- mqc_eigenvalues_at
 - mqc_est, 87
- mqc_eigenvalues_dimension
 - mqc_est, 87
- mqc_eigenvalues_eigenvalues_multiply
 - mqc_est, 88
- mqc_eigenvalues_has_alpha
 - mqc_est, 88
- mqc_eigenvalues_has_beta
 - mqc_est, 88
- mqc_eigenvalues_integral_multiply
 - mqc_est, 88
 - mqc_est::matmul, 122
- mqc_eigenvalues_isallocated
 - mqc_est, 88
- mqc_eigenvalues_output_array
 - mqc_est, 88
 - mqc_est::assignment(=), 113
- mqc_eigenvalues_output_block
 - mqc_est, 89
- mqc_elementmatrixdivide
 - mqc_algebra, 21
 - mqc_algebra::operator(.ewd.), 176
- mqc_elementmatrixproduct
 - mqc_algebra, 22
 - mqc_algebra::operator(.ewp.), 176
- mqc_elementvectorproduct
 - mqc_algebra, 22
 - mqc_algebra::operator(.ewp.), 176
- mqc_eri_integral_contraction
 - mqc_est, 89
 - mqc_est::contraction, 117
- mqc_est, 84
 - gen_det_str, 86
 - get_one_gamma_matrix, 86
 - mqc_build_ci_hamiltonian, 86
 - mqc_eigenvalue_eigenvalue_dotproduct, 86
 - mqc_eigenvalues_add_name, 87
 - mqc_eigenvalues_allocate, 87
 - mqc_eigenvalues_array_name, 87
 - mqc_eigenvalues_array_type, 87
 - mqc_eigenvalues_at, 87
 - mqc_eigenvalues_dimension, 87
 - mqc_eigenvalues_eigenvalues_multiply, 88
 - mqc_eigenvalues_has_alpha, 88
 - mqc_eigenvalues_has_beta, 88
 - mqc_eigenvalues_integral_multiply, 88
 - mqc_eigenvalues_isallocated, 88
 - mqc_eigenvalues_output_array, 88
 - mqc_eigenvalues_output_block, 89
 - mqc_eri_integral_contraction, 89
 - mqc_integral_add_name, 89
 - mqc_integral_allocate, 89
 - mqc_integral_array_name, 89
 - mqc_integral_array_type, 90
 - mqc_integral_at, 90

- mqc_integral_conjugate_transpose, 90
- mqc_integral_delete_energy_list, 90
- mqc_integral_difference, 90
- mqc_integral_dimension, 90
- mqc_integral_eigenvalues_multiply, 91
- mqc_integral_get_energy_list, 91
- mqc_integral_has_alpha, 91
- mqc_integral_has_alphabeta, 91
- mqc_integral_has_beta, 91
- mqc_integral_has_betaalpha, 91
- mqc_integral_identity, 92
- mqc_integral_initialize, 92
- mqc_integral_integral_multiply, 92
- mqc_integral_isallocated, 92
- mqc_integral_matrix_multiply, 92
- mqc_integral_norm, 93
- mqc_integral_output_array, 93
- mqc_integral_output_block, 93
- mqc_integral_output_orbitals, 93
- mqc_integral_scalar_multiply, 93
- mqc_integral_set_energy_list, 94
- mqc_integral_sum, 94
- mqc_integral_swap_orbitals, 94
- mqc_integral_transpose, 94
- mqc_matrix_integral_multiply, 94
- mqc_matrix_spinblockghf, 95
- mqc_matrix_undospinblockghf_eigenvalues, 95
- mqc_matrix_undospinblockghf_integral, 95
- mqc_print_eigenvalues, 95
- mqc_print_integral, 95
- mqc_print_twoeris, 96
- mqc_print_wavefunction, 96
- mqc_scalar_integral_multiply, 96
- mqc_scf_eigenvalues_power, 96
- mqc_scf_integral_contraction, 96
- mqc_scf_integral_determinant, 97
- mqc_scf_integral_diagonalize, 97
- mqc_scf_integral_generalized_eigensystem, 97
- mqc_scf_integral_inverse, 97
- mqc_scf_integral_trace, 97
- mqc_scf_transformation_matrix, 97
- mqc_twoeris_allocate, 98
- mqc_twoeris_at, 98
- slater_condon, 98
- twoeri_trans, 98
- mqc_est::assignment(=), 113
 - mqc_eigenvalues_output_array, 113
 - mqc_integral_output_array, 113
- mqc_est::contraction, 117
 - mqc_eri_integral_contraction, 117
 - mqc_scf_integral_contraction, 118
- mqc_est::dagger, 120
 - mqc_integral_conjugate_transpose, 120
- mqc_est::dot_product, 121
 - mqc_eigenvalue_eigenvalue_dotproduct, 121
- mqc_est::matmul, 122
 - mqc_eigenvalues_eigenvalues_multiply, 122
 - mqc_eigenvalues_integral_multiply, 122
 - mqc_integral_eigenvalues_multiply, 123
 - mqc_integral_integral_multiply, 123
 - mqc_integral_matrix_multiply, 123
 - mqc_matrix_integral_multiply, 123
- mqc_est::mqc_determinant, 126
 - nalpstr, 126
 - nbetstr, 126
 - ndets, 126
 - order, 127
 - strings, 127
- mqc_est::mqc_determinant_string, 127
 - alpha, 127
 - beta, 127
- mqc_est::mqc_matrix_undospinblockghf, 136
 - mqc_matrix_undospinblockghf_eigenvalues, 136
 - mqc_matrix_undospinblockghf_integral, 137
- mqc_est::mqc_print, 139
 - mqc_print_eigenvalues, 139
 - mqc_print_integral, 139
 - mqc_print_twoeris, 140
 - mqc_print_wavefunction, 140
- mqc_est::mqc_pscf_wavefunction, 140
 - nactive, 141
 - ncore, 141
 - nfrz, 141
 - nval, 141
 - pscf_amplitudes, 141
 - pscf_energies, 141
- mqc_est::mqc_scf_eigenvalues, 144
 - addlabel, 144
 - at, 145
 - getblock, 145
 - getlabel, 145
 - power, 145
 - print, 145
- mqc_est::mqc_scf_integral, 146
 - addlabel, 146
 - deleteelist, 146
 - det, 146
 - diag, 146
 - eigensys, 147
 - getblock, 147
 - getelist, 147
 - getlabel, 147
 - identity, 147
 - init, 147
 - inv, 147
 - norm, 148
 - orbitals, 148
 - print, 148

- setelist, [148](#)
 - swap, [148](#)
 - trace, [148](#)
- mqc_est::mqc_twoeris, [149](#)
- print, [150](#)
- mqc_est::mqc_wavefunction, [155](#)
- basis, [156](#)
 - charge, [156](#)
 - core_hamiltonian, [156](#)
 - density_matrix, [157](#)
 - fock_matrix, [157](#)
 - mo_coefficients, [157](#)
 - mo_energies, [157](#)
 - mo_symmetries, [157](#)
 - multiplicity, [157](#)
 - nalpha, [157](#)
 - nbasis, [158](#)
 - nbeta, [158](#)
 - nelectrons, [158](#)
 - overlap_matrix, [158](#)
 - print, [156](#)
 - scf_density_matrix, [158](#)
 - symmetry, [158](#)
 - wf_complex, [158](#)
 - wf_type, [159](#)
- mqc_est::operator(*), [163](#)
- mqc_integral_scalar_multiply, [163](#)
 - mqc_scalar_integral_multiply, [163](#)
- mqc_est::operator(+), [167](#)
- mqc_integral_sum, [167](#)
- mqc_est::operator(-), [170](#)
- mqc_integral_difference, [170](#)
- mqc_est::transpose, [193](#)
- mqc_integral_transpose, [193](#)
- mqc_givens_matrix
- mqc_algebra, [22](#)
- mqc_input_complex_scalar
- mqc_algebra, [22](#)
 - mqc_algebra::assignment(=), [105](#)
- mqc_input_integer_scalar
- mqc_algebra, [23](#)
 - mqc_algebra::assignment(=), [106](#)
- mqc_input_real_scalar
- mqc_algebra, [24](#)
 - mqc_algebra::assignment(=), [107](#)
- mqc_integertscalar
- mqc_algebra, [24](#)
 - mqc_algebra::operator(.gt.), [177](#)
- mqc_integerlescalar
- mqc_algebra, [25](#)
 - mqc_algebra::operator(.le.), [181](#)
- mqc_integerscalaradd
- mqc_algebra, [25](#)
 - mqc_algebra::operator(+), [168](#)
- mqc_integerscalardivide
- mqc_algebra, [25](#)
 - mqc_algebra::operator(/), [187](#)
- mqc_integerscalarmultiply
- mqc_algebra, [26](#)
 - mqc_algebra::operator(*), [160](#)
- mqc_integerscalarsubtract
- mqc_algebra, [26](#)
 - mqc_algebra::operator(-), [171](#)
- mqc_integervectorproduct
- mqc_algebra, [26](#)
 - mqc_algebra::operator(*), [160](#)
- mqc_integral_add_name
- mqc_est, [89](#)
- mqc_integral_allocate
- mqc_est, [89](#)
- mqc_integral_array_name
- mqc_est, [89](#)
- mqc_integral_array_type
- mqc_est, [90](#)
- mqc_integral_at
- mqc_est, [90](#)
- mqc_integral_conjugate_transpose
- mqc_est, [90](#)
 - mqc_est::dagger, [120](#)
- mqc_integral_delete_energy_list
- mqc_est, [90](#)
- mqc_integral_difference
- mqc_est, [90](#)
 - mqc_est::operator(-), [170](#)
- mqc_integral_dimension
- mqc_est, [90](#)
- mqc_integral_eigenvalues_multiply
- mqc_est, [91](#)
 - mqc_est::matmul, [123](#)
- mqc_integral_get_energy_list
- mqc_est, [91](#)
- mqc_integral_has_alpha
- mqc_est, [91](#)
- mqc_integral_has_alphabeta
- mqc_est, [91](#)
- mqc_integral_has_beta
- mqc_est, [91](#)
- mqc_integral_has_betaalpha
- mqc_est, [91](#)
- mqc_integral_identity
- mqc_est, [92](#)
- mqc_integral_initialize
- mqc_est, [92](#)
- mqc_integral_integral_multiply
- mqc_est, [92](#)
 - mqc_est::matmul, [123](#)
- mqc_integral_isallocated
- mqc_est, [92](#)

mqc_integral_matrix_multiply
 mqc_est, 92
 mqc_est::matmul, 123
 mqc_integral_norm
 mqc_est, 93
 mqc_integral_output_array
 mqc_est, 93
 mqc_est::assignment(=), 113
 mqc_integral_output_block
 mqc_est, 93
 mqc_integral_output_orbitals
 mqc_est, 93
 mqc_integral_scalar_multiply
 mqc_est, 93
 mqc_est::operator(*), 163
 mqc_integral_set_energy_list
 mqc_est, 94
 mqc_integral_sum
 mqc_est, 94
 mqc_est::operator(+), 167
 mqc_integral_swap_orbitals
 mqc_est, 94
 mqc_integral_transpose
 mqc_est, 94
 mqc_est::transpose, 193
 mqc_length_vector
 mqc_algebra, 26
 mqc_matrix_cast_complex
 mqc_algebra, 26
 mqc_algebra::mqc_cast_complex, 125
 mqc_matrix_cast_real
 mqc_algebra, 26
 mqc_algebra::mqc_cast_real, 125
 mqc_matrix_columns
 mqc_algebra, 27
 mqc_matrix_conjugate_transpose
 mqc_algebra, 27
 mqc_algebra::dagger, 119
 mqc_matrix_copy_complex2int
 mqc_algebra, 27
 mqc_matrix_copy_complex2real
 mqc_algebra, 27
 mqc_matrix_copy_int2complex
 mqc_algebra, 27
 mqc_matrix_copy_int2real
 mqc_algebra, 27
 mqc_matrix_copy_real2complex
 mqc_algebra, 28
 mqc_matrix_copy_real2int
 mqc_algebra, 28
 mqc_matrix_determinant
 mqc_algebra, 28
 mqc_matrix_diag2full
 mqc_algebra, 28
 mqc_matrix_diag2symm
 mqc_algebra, 28
 mqc_matrix_diagmatrix_put_complex
 mqc_algebra, 28
 mqc_algebra::mqc_matrix_diagmatrix_put, 134
 mqc_matrix_diagmatrix_put_integer
 mqc_algebra, 29
 mqc_algebra::mqc_matrix_diagmatrix_put, 135
 mqc_matrix_diagmatrix_put_real
 mqc_algebra, 29
 mqc_algebra::mqc_matrix_diagmatrix_put, 135
 mqc_matrix_diagmatrix_put_vector
 mqc_algebra, 29
 mqc_algebra::mqc_matrix_diagmatrix_put, 135
 mqc_matrix_diagonalize
 mqc_algebra, 29
 mqc_matrix_full2diag
 mqc_algebra, 29
 mqc_matrix_full2symm
 mqc_algebra, 29
 mqc_matrix_generalized_eigensystem
 mqc_algebra, 30
 mqc_matrix_havecomplex
 mqc_algebra, 30
 mqc_algebra::mqc_have_complex, 128
 mqc_matrix_havediagonal
 mqc_algebra, 30
 mqc_matrix_havefull
 mqc_algebra, 30
 mqc_matrix_haveinteger
 mqc_algebra, 30
 mqc_algebra::mqc_have_int, 128
 mqc_matrix_havereal
 mqc_algebra, 30
 mqc_algebra::mqc_have_real, 129
 mqc_matrix_havesymmetric
 mqc_algebra, 31
 mqc_matrix_identity
 mqc_algebra, 31
 mqc_matrix_initialize
 mqc_algebra, 31
 mqc_matrix_integral_multiply
 mqc_est, 94
 mqc_est::matmul, 123
 mqc_matrix_inverse
 mqc_algebra, 31
 mqc_matrix_isallocated
 mqc_algebra, 31
 mqc_matrix_matrix_at
 mqc_algebra, 31
 mqc_matrix_matrix_contraction
 mqc_algebra, 33
 mqc_algebra::contraction, 117
 mqc_matrix_matrix_put

- mqc_algebra, 33
- mqc_matrix_norm
 - mqc_algebra, 33
- mqc_matrix_rms_max
 - mqc_algebra, 34
- mqc_matrix_rows
 - mqc_algebra, 34
- mqc_matrix_scalar_at
 - mqc_algebra, 34
- mqc_matrix_scalar_put
 - mqc_algebra, 34
- mqc_matrix_set
 - mqc_algebra, 34
- mqc_matrix_spinblockghf
 - mqc_est, 95
- mqc_matrix_sqrt
 - mqc_algebra, 35
- mqc_matrix_storagetype
 - mqc_algebra, 35
- mqc_matrix_svd
 - mqc_algebra, 35
- mqc_matrix_symm2diag
 - mqc_algebra, 35
- mqc_matrix_symm2full
 - mqc_algebra, 35
- mqc_matrix_symm2full_func
 - mqc_algebra, 36
- mqc_matrix_symmetrize
 - mqc_algebra, 36
- mqc_matrix_symmmatrix_put_complex
 - mqc_algebra, 36
 - mqc_algebra::mqc_matrix_symmmatrix_put, 136
- mqc_matrix_symmmatrix_put_integer
 - mqc_algebra, 36
 - mqc_algebra::mqc_matrix_symmmatrix_put, 136
- mqc_matrix_symmmatrix_put_real
 - mqc_algebra, 36
 - mqc_algebra::mqc_matrix_symmmatrix_put, 136
- mqc_matrix_symmsymmr4tensor_put_complex
 - mqc_algebra, 36
- mqc_matrix_symmsymmr4tensor_put_real
 - mqc_algebra, 37
- mqc_matrix_test_diagonal
 - mqc_algebra, 37
- mqc_matrix_test_symmetric
 - mqc_algebra, 37
- mqc_matrix_trace
 - mqc_algebra, 37
- mqc_matrix_transpose
 - mqc_algebra, 37
 - mqc_algebra::transpose, 193
- mqc_matrix_undospinblockghf_eigenvalues
 - mqc_est, 95
 - mqc_est::mqc_matrix_undospinblockghf, 136
- mqc_matrix_undospinblockghf_integral
 - mqc_est, 95
 - mqc_est::mqc_matrix_undospinblockghf, 137
- mqc_matrix_vector_at
 - mqc_algebra, 37
- mqc_matrix_vector_put
 - mqc_algebra, 38
- mqc_matrixmatrixdotproduct
 - mqc_algebra, 38
 - mqc_algebra::matmul, 121
 - mqc_algebra::operator(.dot.), 174
- mqc_matrixmatrixproduct
 - mqc_algebra, 38
 - mqc_algebra::operator(*), 160
- mqc_matrixmatrixsubtract
 - mqc_algebra, 38
 - mqc_algebra::operator(-), 171
- mqc_matrixmatrixsum
 - mqc_algebra, 38
 - mqc_algebra::operator(+), 168
- mqc_matrixscalarproduct
 - mqc_algebra, 39
 - mqc_algebra::operator(*), 160
- mqc_matrixvectordotproduct
 - mqc_algebra, 39
 - mqc_algebra::matmul, 121
 - mqc_algebra::operator(.dot.), 174
- mqc_outer
 - mqc_algebra, 39
 - mqc_algebra::operator(.outer.), 186
- mqc_output_complex_scalar
 - mqc_algebra, 39
 - mqc_algebra::assignment(=), 107
- mqc_output_integer_scalar
 - mqc_algebra, 40
 - mqc_algebra::assignment(=), 108
- mqc_output_mqcscalar_scalar
 - mqc_algebra, 41
 - mqc_algebra::assignment(=), 109
- mqc_output_real_scalar
 - mqc_algebra, 41
 - mqc_algebra::assignment(=), 109
- mqc_print_eigenvalues
 - mqc_est, 95
 - mqc_est::mqc_print, 139
- mqc_print_integral
 - mqc_est, 95
 - mqc_est::mqc_print, 139
- mqc_print_matrix_algebra1
 - mqc_algebra, 42
 - mqc_algebra::mqc_print, 137
- mqc_print_r4tensor_algebra1
 - mqc_algebra, 42
 - mqc_algebra::mqc_print, 137

mqc_print_scalar_algebra1
 mqc_algebra, 42
 mqc_algebra::mqc_print, 138
 mqc_print_twoeris
 mqc_est, 96
 mqc_est::mqc_print, 140
 mqc_print_vector_algebra1
 mqc_algebra, 43
 mqc_algebra::mqc_print, 139
 mqc_print_wavefunction
 mqc_est, 96
 mqc_est::mqc_print, 140
 mqc_r4tensor_at
 mqc_algebra, 44
 mqc_r4tensor_havecomplex
 mqc_algebra, 44
 mqc_r4tensor_haveinteger
 mqc_algebra, 44
 mqc_r4tensor_havereal
 mqc_algebra, 44
 mqc_r4tensor_initialize
 mqc_algebra, 44
 mqc_r4tensor_put
 mqc_algebra, 45
 mqc_realgtscalar
 mqc_algebra, 45
 mqc_algebra::operator(.gt.), 178
 mqc_reallscalar
 mqc_algebra, 45
 mqc_algebra::operator(.le.), 181
 mqc_realltscalar
 mqc_algebra, 45
 mqc_algebra::operator(.lt.), 182
 mqc_realscalaradd
 mqc_algebra, 46
 mqc_algebra::operator(+), 168
 mqc_realscalardivide
 mqc_algebra, 46
 mqc_algebra::operator(/), 187
 mqc_realscalarmultiply
 mqc_algebra, 47
 mqc_algebra::operator(*), 160
 mqc_realscalarsubtract
 mqc_algebra, 47
 mqc_algebra::operator(-), 172
 mqc_realvectorproduct
 mqc_algebra, 47
 mqc_algebra::operator(*), 161
 mqc_scalar_acos
 mqc_algebra, 47
 mqc_algebra::acos, 102
 mqc_scalar_asin
 mqc_algebra, 48
 mqc_algebra::asin, 104
 mqc_scalar_atan
 mqc_algebra, 48
 mqc_algebra::atan, 113
 mqc_scalar_atan2
 mqc_algebra, 49
 mqc_algebra::atan2, 114
 mqc_scalar_cmplx
 mqc_algebra, 50
 mqc_algebra::cmplx, 115
 mqc_scalar_complex_conjugate
 mqc_algebra, 50
 mqc_algebra::conjg, 116
 mqc_scalar_complex_imagpart
 mqc_algebra, 50
 mqc_algebra::aimag, 103
 mqc_scalar_complex_realpart
 mqc_algebra, 51
 mqc_algebra::real, 189
 mqc_scalar_cos
 mqc_algebra, 51
 mqc_algebra::cos, 118
 mqc_scalar_get_abs_value
 mqc_algebra, 51
 mqc_algebra::abs, 101
 mqc_scalar_get_intrinsic_complex
 mqc_algebra, 52
 mqc_scalar_get_intrinsic_integer
 mqc_algebra, 53
 mqc_scalar_get_intrinsic_real
 mqc_algebra, 53
 mqc_scalar_get_random_value
 mqc_algebra, 54
 mqc_scalar_havecomplex
 mqc_algebra, 54
 mqc_scalar_haveinteger
 mqc_algebra, 55
 mqc_scalar_havereal
 mqc_algebra, 56
 mqc_scalar_integral_multiply
 mqc_est, 96
 mqc_est::operator(*), 163
 mqc_scalar_isallocated
 mqc_algebra, 56
 mqc_scalar_sin
 mqc_algebra, 57
 mqc_algebra::sin, 190
 mqc_scalar_sqrt
 mqc_algebra, 58
 mqc_algebra::sqrt, 191
 mqc_scalar_tan
 mqc_algebra, 58
 mqc_algebra::tan, 192
 mqc_scalaradd
 mqc_algebra, 59

- mqc_algebra::operator(+), 169
- mqc_scalarcomplexadd
 - mqc_algebra, 59
 - mqc_algebra::operator(+), 169
- mqc_scalarcomplexdivide
 - mqc_algebra, 60
 - mqc_algebra::operator(/), 187
- mqc_scalarcomplexexponent
 - mqc_algebra, 60
 - mqc_algebra::operator(**), 164
- mqc_scalarcomplexmultiply
 - mqc_algebra, 60
 - mqc_algebra::operator(*), 161
- mqc_scalarcomplexsubtract
 - mqc_algebra, 61
 - mqc_algebra::operator(-), 172
- mqc_scalardivide
 - mqc_algebra, 61
 - mqc_algebra::operator(/), 187
- mqc_scalareq
 - mqc_algebra, 61
 - mqc_algebra::operator(.eq.), 175
- mqc_scalarexponent
 - mqc_algebra, 62
 - mqc_algebra::operator(**), 165
- mqc_scalarge
 - mqc_algebra, 63
 - mqc_algebra::operator(.ge.), 177
- mqc_scalargt
 - mqc_algebra, 63
 - mqc_algebra::operator(.gt.), 178
- mqc_scalargtinteger
 - mqc_algebra, 64
 - mqc_algebra::operator(.gt.), 179
- mqc_scalargtreal
 - mqc_algebra, 65
 - mqc_algebra::operator(.gt.), 180
- mqc_scalarintegeradd
 - mqc_algebra, 65
 - mqc_algebra::operator(+), 169
- mqc_scalarintegerdivide
 - mqc_algebra, 65
 - mqc_algebra::operator(/), 188
- mqc_scalarintegerexponent
 - mqc_algebra, 65
 - mqc_algebra::operator(**), 166
- mqc_scalarintegermultiply
 - mqc_algebra, 66
 - mqc_algebra::operator(*), 161
- mqc_scalarintegersubtract
 - mqc_algebra, 66
 - mqc_algebra::operator(-), 172
- mqc_scalarle
 - mqc_algebra, 66
 - mqc_algebra::operator(.le.), 181
- mqc_scalarleinteger
 - mqc_algebra, 66
 - mqc_algebra::operator(.le.), 181
- mqc_scalarlereal
 - mqc_algebra, 67
 - mqc_algebra::operator(.le.), 181
- mqc_scalarlt
 - mqc_algebra, 67
 - mqc_algebra::operator(.lt.), 183
- mqc_scalarltreal
 - mqc_algebra, 68
 - mqc_algebra::operator(.lt.), 183
- mqc_scalarmatrixproduct
 - mqc_algebra, 68
 - mqc_algebra::operator(*), 161
- mqc_scalarmultiply
 - mqc_algebra, 68
 - mqc_algebra::operator(*), 161
- mqc_scalarne
 - mqc_algebra, 69
 - mqc_algebra::operator(.ne.), 185
- mqc_scalarrealadd
 - mqc_algebra, 70
 - mqc_algebra::operator(+), 170
- mqc_scalarrealddivide
 - mqc_algebra, 70
 - mqc_algebra::operator(/), 188
- mqc_scalarrealexponent
 - mqc_algebra, 70
 - mqc_algebra::operator(**), 166
- mqc_scalarrealmultiply
 - mqc_algebra, 71
 - mqc_algebra::operator(*), 162
- mqc_scalarrealsubtract
 - mqc_algebra, 71
 - mqc_algebra::operator(-), 172
- mqc_scalarsubtract
 - mqc_algebra, 71
 - mqc_algebra::operator(-), 172
- mqc_scalarvectordifference
 - mqc_algebra, 72
 - mqc_algebra::operator(-), 173
- mqc_scalarvectorproduct
 - mqc_algebra, 72
 - mqc_algebra::operator(*), 162
- mqc_scalarvectorsum
 - mqc_algebra, 72
 - mqc_algebra::operator(+), 170
- mqc_scf_eigenvalues_power
 - mqc_est, 96
- mqc_scf_integral_contraction
 - mqc_est, 96
 - mqc_est::contraction, 118

mqc_scf_integral_determinant
 mqc_est, 97
 mqc_scf_integral_diagonalize
 mqc_est, 97
 mqc_scf_integral_generalized_eigensystem
 mqc_est, 97
 mqc_scf_integral_inverse
 mqc_est, 97
 mqc_scf_integral_trace
 mqc_est, 97
 mqc_scf_transformation_matrix
 mqc_est, 97
 mqc_set_array2tensor
 mqc_algebra, 72
 mqc_algebra::assignment(=), 110
 mqc_set_array2vector_complex
 mqc_algebra, 73
 mqc_algebra::assignment(=), 110
 mqc_algebra::mqc_set_array2vector, 149
 mqc_set_array2vector_integer
 mqc_algebra, 73
 mqc_algebra::assignment(=), 110
 mqc_algebra::mqc_set_array2vector, 149
 mqc_set_array2vector_real
 mqc_algebra, 73
 mqc_algebra::assignment(=), 110
 mqc_algebra::mqc_set_array2vector, 149
 mqc_set_complexarray2matrix
 mqc_algebra, 73
 mqc_algebra::assignment(=), 111
 mqc_set_integerarray2matrix
 mqc_algebra, 73
 mqc_algebra::assignment(=), 111
 mqc_set_matrix2complexarray
 mqc_algebra, 73
 mqc_algebra::assignment(=), 111
 mqc_set_matrix2integerarray
 mqc_algebra, 74
 mqc_algebra::assignment(=), 111
 mqc_set_matrix2matrix
 mqc_algebra, 74
 mqc_algebra::assignment(=), 111
 mqc_set_matrix2realarray
 mqc_algebra, 74
 mqc_algebra::assignment(=), 111
 mqc_set_realarray2matrix
 mqc_algebra, 74
 mqc_algebra::assignment(=), 112
 mqc_set_vector2complexarray
 mqc_algebra, 74
 mqc_algebra::assignment(=), 112
 mqc_set_vector2integerarray
 mqc_algebra, 74
 mqc_algebra::assignment(=), 112
 mqc_set_vector2realarray
 mqc_algebra, 75
 mqc_algebra::assignment(=), 112
 mqc_set_vector2vector
 mqc_algebra, 75
 mqc_algebra::assignment(=), 112
 mqc_twoeris_allocate
 mqc_est, 98
 mqc_twoeris_at
 mqc_est, 98
 mqc_vector2diagmatrix
 mqc_algebra, 75
 mqc_vector_abs
 mqc_algebra, 75
 mqc_algebra::abs, 102
 mqc_vector_arg sort
 mqc_algebra, 75
 mqc_vector_cast_complex
 mqc_algebra, 75
 mqc_algebra::mqc_cast_complex, 125
 mqc_vector_cast_real
 mqc_algebra, 76
 mqc_algebra::mqc_cast_real, 126
 mqc_vector_cmplx
 mqc_algebra, 76
 mqc_algebra::cmplx, 116
 mqc_vector_complex_imagpart
 mqc_algebra, 76
 mqc_algebra::aimag, 103
 mqc_vector_complex_realpart
 mqc_algebra, 76
 mqc_algebra::real, 189
 mqc_vector_conjugate_transpose
 mqc_algebra, 76
 mqc_algebra::dagger, 119
 mqc_vector_copy_complex2int
 mqc_algebra, 76
 mqc_vector_copy_complex2real
 mqc_algebra, 77
 mqc_vector_copy_int2complex
 mqc_algebra, 77
 mqc_vector_copy_int2real
 mqc_algebra, 77
 mqc_vector_copy_real2complex
 mqc_algebra, 77
 mqc_vector_copy_real2int
 mqc_algebra, 77
 mqc_vector_havecomplex
 mqc_algebra, 77
 mqc_algebra::mqc_have_complex, 128
 mqc_vector_haveinteger
 mqc_algebra, 78
 mqc_algebra::mqc_have_int, 129
 mqc_vector_havereal

- mqc_algebra, 78
- mqc_algebra::mqc_have_real, 129
- mqc_vector_initialize
 - mqc_algebra, 78
- mqc_vector_isallocated
 - mqc_algebra, 78
- mqc_vector_iscolumn
 - mqc_algebra, 78
- mqc_vector_maxloc
 - mqc_algebra, 78
- mqc_vector_maxval
 - mqc_algebra, 79
- mqc_vector_minloc
 - mqc_algebra, 79
- mqc_vector_minval
 - mqc_algebra, 79
- mqc_vector_norm
 - mqc_algebra, 79
- mqc_vector_pop
 - mqc_algebra, 79
- mqc_vector_power
 - mqc_algebra, 79
- mqc_vector_push
 - mqc_algebra, 80
- mqc_vector_scalar_at
 - mqc_algebra, 80
- mqc_vector_scalar_increment
 - mqc_algebra, 80
- mqc_vector_scalar_put
 - mqc_algebra, 80
- mqc_vector_shift
 - mqc_algebra, 80
- mqc_vector_sort
 - mqc_algebra, 80
- mqc_vector_sqrt
 - mqc_algebra, 81
- mqc_vector_transpose
 - mqc_algebra, 81
 - mqc_algebra::transpose, 194
- mqc_vector_unshift
 - mqc_algebra, 81
- mqc_vector_vector_at
 - mqc_algebra, 81
- mqc_vector_vector_put
 - mqc_algebra, 81
- mqc_vectorcomplexdivide
 - mqc_algebra, 81
 - mqc_algebra::operator(/), 188
- mqc_vectorcomplexproduct
 - mqc_algebra, 82
 - mqc_algebra::operator(*), 162
- mqc_vectorintegerdivide
 - mqc_algebra, 82
 - mqc_algebra::operator(/), 188
- mqc_vectorintegerproduct
 - mqc_algebra, 82
 - mqc_algebra::operator(*), 162
- mqc_vectormatrixdotproduct
 - mqc_algebra, 82
 - mqc_algebra::matmul, 122
 - mqc_algebra::operator(.dot.), 174
- mqc_vectorrealdive
 - mqc_algebra, 82
 - mqc_algebra::operator(/), 189
- mqc_vectorrealproduct
 - mqc_algebra, 82
 - mqc_algebra::operator(*), 163
- mqc_vectorscalardivide
 - mqc_algebra, 83
 - mqc_algebra::operator(/), 189
- mqc_vectorscalarproduct
 - mqc_algebra, 83
 - mqc_algebra::operator(*), 163
- mqc_vectorvectordifference
 - mqc_algebra, 83
 - mqc_algebra::operator(-), 173
- mqc_vectorvectordotproduct
 - mqc_algebra, 83
 - mqc_algebra::dot_product, 120
 - mqc_algebra::operator(.dot.), 174
- mqc_vectorvectorsum
 - mqc_algebra, 83
 - mqc_algebra::operator(+), 170
- multiplicity
 - mqc_est::mqc_wavefunction, 157
- nactive
 - mqc_est::mqc_pscf_wavefunction, 141
- nalpha
 - mqc_est::mqc_wavefunction, 157
- nalpstr
 - mqc_est::mqc_determinant, 126
- nbasis
 - mqc_est::mqc_wavefunction, 158
- nbeta
 - mqc_est::mqc_wavefunction, 158
- nbetstr
 - mqc_est::mqc_determinant, 126
- ncore
 - mqc_est::mqc_pscf_wavefunction, 141
- ndets
 - mqc_est::mqc_determinant, 126
- nelectrons
 - mqc_est::mqc_wavefunction, 158
- nfrz
 - mqc_est::mqc_pscf_wavefunction, 141
- norm
 - mqc_algebra::mqc_matrix, 132

- mqc_algebra::mqc_vector, 152
 - mqc_est::mqc_scf_integral, 148
- nval
 - mqc_est::mqc_pscf_wavefunction, 141
- orbitals
 - mqc_est::mqc_scf_integral, 148
- order
 - mqc_est::mqc_determinant, 127
- overlap_matrix
 - mqc_est::mqc_wavefunction, 158
- pop
 - mqc_algebra::mqc_vector, 152
- power
 - mqc_algebra::mqc_vector, 153
 - mqc_est::mqc_scf_eigenvalues, 145
- print
 - mqc_algebra::mqc_matrix, 132
 - mqc_algebra::mqc_r4tensor, 142
 - mqc_algebra::mqc_scalar, 144
 - mqc_algebra::mqc_vector, 153
 - mqc_est::mqc_scf_eigenvalues, 145
 - mqc_est::mqc_scf_integral, 148
 - mqc_est::mqc_twoeris, 150
 - mqc_est::mqc_wavefunction, 156
- pscf_amplitudes
 - mqc_est::mqc_pscf_wavefunction, 141
- pscf_energies
 - mqc_est::mqc_pscf_wavefunction, 141
- push
 - mqc_algebra::mqc_vector, 153
- put
 - mqc_algebra::mqc_matrix, 132
 - mqc_algebra::mqc_r4tensor, 143
 - mqc_algebra::mqc_vector, 153
- random
 - mqc_algebra::mqc_scalar, 144
- rmsmax
 - mqc_algebra::mqc_matrix, 132
- rval
 - mqc_algebra::mqc_scalar, 144
- s_type
 - mqc_algebra::mqc_matrix, 132
- scf_density_matrix
 - mqc_est::mqc_wavefunction, 158
- set
 - mqc_algebra::mqc_matrix, 133
- setelist
 - mqc_est::mqc_scf_integral, 148
- shift
 - mqc_algebra::mqc_vector, 153
- size
 - mqc_algebra::mqc_vector, 153
- slater_condon
 - mqc_est, 98
- sort
 - mqc_algebra::mqc_vector, 153
- sqrt
 - mqc_algebra::mqc_matrix, 133
 - mqc_algebra::mqc_vector, 154
- src/mqc_algebra.F03, 195
- src/mqc_est.F03, 202
- strings
 - mqc_est::mqc_determinant, 127
- svd
 - mqc_algebra::mqc_matrix, 133
- swap
 - mqc_est::mqc_scf_integral, 148
- symindexhash
 - mqc_algebra, 83
- symmetry
 - mqc_est::mqc_wavefunction, 158
- trace
 - mqc_algebra::mqc_matrix, 133
 - mqc_est::mqc_scf_integral, 148
- transpose
 - mqc_algebra::mqc_matrix, 133
 - mqc_algebra::mqc_vector, 154
- twoeri_trans
 - mqc_est, 98
- unshift
 - mqc_algebra::mqc_vector, 154
- vat
 - mqc_algebra::mqc_matrix, 133
 - mqc_algebra::mqc_vector, 154
- vecc
 - mqc_algebra::mqc_vector, 155
- veci
 - mqc_algebra::mqc_vector, 155
- vecr
 - mqc_algebra::mqc_vector, 155
- vput
 - mqc_algebra::mqc_matrix, 133
 - mqc_algebra::mqc_vector, 154
- wf_complex
 - mqc_est::mqc_wavefunction, 158
- wf_type
 - mqc_est::mqc_wavefunction, 159