MQCPack

Generated by Doxygen 1.8.16

Chapter 1

Modules Index

1.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 2

Data Type Index

2.1 Class Hierarchy

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

Chapter 3

Data Type Index

3.1 Data Types List

Here are the data types with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 mqc_algebra Module Reference

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

Data Types

- interface abs

  Takes the absolute value
- interface acos

  Returns the arccosine
- interface aimag

  Returns the imaginary part
- interface asin

  Returns the arcsine
- interface assignment(=)

  Assigns a variable to the value of another
- interface atan

  Returns the arctangent
- interface atan2

  Returns the arctangent accounting for circle quadrant
- interface cmplx

  Defines a complex number
- interface conjg

  Returns the complex conjugate
- interface contraction

  Contracts two arrays
- interface cos

  Returns the cosine
- interface dagger

  Returns the Hermitian conjugate

- interface dot_product

  Returns the dot product
- interface matmul

  Multiplies two arrays
- interface matrix_symm2sq

  Sets a symmetric packed intrinsic array as a square packed intrinsic array
- interface mqc_cast_complex

  Sets an array to complex type
- interface mqc_cast_integer

  Sets an array to integer type
- interface mqc_cast_real

  Sets an array to real type
- interface mqc_have_complex

  Determines in an array is complex type
- interface mqc_have_int

  Determines in an array is integer type
- interface mqc_have_real

  Determines in an array is real type
- type mqc_matrix

  Rank 2 array variable
- interface mqc_matrix_diagmatrix_put

  Sets a diagonal packed intinsic array as an MQC Matrix object
- interface mqc_matrix_symmmatrix_put

  Sets a symmetric packed intrinsic array as an MQC Matrix object
- interface mqc_print

  Prints an object
- type mqc_r4tensor

  Updates the specified element of the MQC Matrix to the specified value
- type mqc_scalar

  Rank 0 array variable
- interface mqc_set_array2vector

  Sets an intrinsic array as an MQC Algebra object
- type mqc_vector

  Rank 1 array variable
- interface operator(∗)

  Multiplies two variables
- interface operator(∗∗)

  Exponentials a variable to the power of another
- interface operator(+)

  Sums two variables
- interface operator(-)

  Subtracts two variables
- interface operator(.dot.)

  Computes the inner product of two arrays
- interface operator(.eq.)

  Determines if two variables are equal
- interface operator(.ewd.)

Computes the element-wise quotient of two arrays
- interface operator(.ewp.)

  Computes the element-wise product of two arrays
- interface operator(.ge.)

  Determines if a variable is greater than or equal to another
- interface operator(.gt.)

  Determines if a variable is greater than another
- interface operator(.le.)

  Determines if a variable is less than or equal to another
- interface operator(.lt.)

  Determines if a variable is less than another
- interface operator(.ne.)

  Determines if two variables are not equal
- interface operator(.outer.)

  Computes the outer product of two vectors
- interface operator(.x.)

  Computes the cross product of two vectors
- interface operator(/)

  Divides two variables
- interface real

  Returns the real part
- interface sin

  Returns the sine
- interface sqrt

  Returns the square root
- interface tan

  Returns the tangent
- interface transpose

  Returns the transpose

Functions/Subroutines

- integer(kind=int64) function factorial (n)

  Factorial returns the factorial of an integer
- integer(kind=int64) function bin_coeff (N, K)

  Bin_Coeff returns the binomial coefficient of (n,k)
- subroutine mqc_allocate_scalar (Scalar, Data_type)

  MQC_Allocate_Scalar is used to allocate a scalar type variable of the MQC_Scalar class
- subroutine mqc_deallocate_scalar (Scalar)

  MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class
- logical function mqc_scalar_isallocated (Scalar)

  MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar
- subroutine mqc_input_integer_scalar (ScalarOut, ScalarIn)

  MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an MQC_Scalar
- subroutine mqc_input_real_scalar (ScalarOut, ScalarIn)

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

- subroutine mqc_input_complex_scalar (ScalarOut, ScalarIn)

  MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an MQC_Scalar
- subroutine mqc_output_mqcscalar_scalar (ScalarOut, ScalarIn)

  MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar
- subroutine mqc_output_integer_scalar (ScalarOut, ScalarIn)

  MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar
- subroutine mqc_output_real_scalar (ScalarOut, ScalarIn)

  MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar
- subroutine mqc_output_complex_scalar (ScalarOut, ScalarIn)

  MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar
- subroutine mqc_print_scalar_algebra1 (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)

  MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar
- type(mqc_scalar) function mqc_scalar_cmplx (Scalar1, Scalar2)

  MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars
- type(mqc_scalar) function mqc_scalar_sqrt (Scalar)

  MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar
- type(mqc_scalar) function mqc_scalar_sin (Scalar)

  MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar
- type(mqc_scalar) function mqc_scalar_cos (Scalar)

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

  MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar
- type(mqc_scalar) function mqc_scalar_asin (Scalar)

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

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

  MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar
- type(mqc_scalar) function mqc_scalar_atan2 (Scalar)

  MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram
- logical function mqc_scalar_havereal (Scalar)

  MQC_Scalar_HaveReal is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real
- logical function mqc_scalar_haveinteger (Scalar)

  MQC_Scalar_HaveInteger is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer
- logical function mqc_scalar_havecomplex (Scalar)

  MQC_Scalar_HaveComplex is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex
- real(kind=real64) function mqc_scalar_get_intrinsic_real (Scalar)

  MQC_Scalar_Get_Intrinsic_Real is a function that returns the MQC_scalar value as an intrinsic real
- integer(kind=int64) function mqc_scalar_get_intrinsic_integer (Scalar)

  MQC_Scalar_Get_Intrinsic_Integer is a function that returns the MQC_scalar value as an intrinsic integer
- complex(kind=real64) function mqc_scalar_get_intrinsic_complex (Scalar)

  MQC_Scalar_Get_Intrinsic_Complex is a function that returns the MQC_scalar value as an intrinsic complex
- type(mqc_scalar) function mqc_scalar_get_abs_value (Scalar)

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

- subroutine mqc_scalar_get_random_value (Scalar, Seed, Distribution)
  MQC_Scalar_Get_Random_Value is a function that returns a random real value from a specified distribution
- type(mqc_scalar) function mqc_scalaradd (Scalar1, Scalar2)
  MQC_ScalarAdd is a function that sums two MQC_Scalar objects
- type(mqc_scalar) function mqc_scalarsubtract (Scalar1, Scalar2)
  MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects
- type(mqc_scalar) function mqc_scalarmultiply (Scalar1, Scalar2)
  MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects
- type(mqc_scalar) function mqc_scalardivide (Scalar1, Scalar2)
  MQC_ScalarDivide is a function that divides two MQC_Scalar objects
- type(mqc_scalar) function mqc_scalarexponent (Scalar1, Scalar2)
  MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegerexponent (Scalar, 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, CompIn)
  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_integergtscalar (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)
  MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar
- logical function mqc_scalargtreal (Scalar, RealIn)
  MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real
- logical function mqc_scalarle (Scalar1, Scalar2)
  MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar
- logical function mqc_reallescalar (RealIn, Scalar)
  MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar
- logical function mqc_scalarlereal (Scalar, RealIn)
  MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real
- logical function mqc_integerlescalar (IntIn, Scalar)
  MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar
- logical function mqc_scalarleinteger (Scalar, IntIn)

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

- logical function mqc_scalarge (Scalar1, Scalar2)

  MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_↩ Scalar
- type(mqc_scalar) function mqc_scalar_complex_conjugate (ScalarIn)

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

  MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar
- type(mqc_scalar) function mqc_scalar_complex_imagpart (ScalarIn)

  MQC_Scalar_Complex_ImagPart is a function that returns the inaginary part of an MQC_Scalar
- type(mqc_scalar) function mqc_integerscalarmultiply (IntegerIn, Scalar)

  MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegermultiply (Scalar, IntegerIn)

  MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_realscalarmultiply (RealIn, Scalar)

  MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealmultiply (Scalar, RealIn)

  MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_complexscalarmultiply (ComplexIn, Scalar)

  MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexmultiply (Scalar, ComplexIn)

  MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_integerscalardivide (IntegerIn, Scalar)

  MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegerdivide (Scalar, IntegerIn)

  MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer
- type(mqc_scalar) function mqc_realscalardivide (RealIn, Scalar)

  MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealdivide (Scalar, RealIn)

  MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an intrinsic real
- type(mqc_scalar) function mqc_complexscalardivide (ComplexIn, Scalar)

  MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexdivide (Scalar, ComplexIn)

  MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex
- type(mqc_scalar) function mqc_integerscalaradd (IntegerIn, Scalar)

  MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegeradd (Scalar, IntegerIn)

  MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_realscalaradd (RealIn, Scalar)

  MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealadd (Scalar, RealIn)

  MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_complexscalaradd (ComplexIn, Scalar)

  MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexadd (Scalar, ComplexIn)

  MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_integerscalarsubtract (IntegerIn, Scalar)

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

- type(mqc_scalar) function mqc_scalarintegersubtract (Scalar, IntegerIn)

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

- type(mqc_scalar) function mqc_realscalarsubtract (RealIn, Scalar)

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

- type(mqc_scalar) function mqc_scalarrealsubtract (Scalar, RealIn)

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

- type(mqc_scalar) function mqc_complexscalarsubtract (ComplexIn, Scalar)

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

- type(mqc_scalar) function mqc_scalarcomplexsubtract (Scalar, ComplexIn)

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

- subroutine mqc_allocate_vector (N, Vector, Data_Type)

  MQC_Allocate_Vector is used to allocate a vector type variable of the MQC_Vector class

- subroutine mqc_deallocate_vector (Vector)

  MQC_Deallocate_Vector is used to deallocate a vector type variable of the MQC_Vector class

- integer(kind=int64) function mqc_length_vector (Vector)

  MQC_Length_Vector is used to return the length of an MQC vector

- logical function mqc_vector_havereal (Vector)

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

- logical function mqc_vector_haveinteger (Vector)

  MQC_Vector_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector

- logical function mqc_vector_havecomplex (Vector)

  MQC_Vector_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector

- logical function mqc_vector_iscolumn (Vector)

  MQC_Vector_IsColumn is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector

- subroutine mqc_vector_copy_int2real (Vector)

  MQC_Vector_Copy_Int2Real is a subroutine that copies an integer MQC_Vector into its real vector

- subroutine mqc_vector_copy_int2complex (Vector)

  MQC_Vector_Copy_Int2Complex is a subroutine that copies an integer MQC_Vector into its complex vector

- subroutine mqc_vector_copy_real2int (Vector)

  MQC_Vector_Copy_Real2Int is a subroutine that copies a real MQC_Vector into its integer vector

- subroutine mqc_vector_copy_real2complex (Vector)

  MQC_Vector_Copy_Real2Complex is a subroutine that copies a real MQC_Vector into its complex vector

- subroutine mqc_vector_copy_complex2int (Vector)

  MQC_Vector_Copy_Complex2Int is a subroutine that copies a complex MQC_Vector into its integer vector

- subroutine mqc_vector_copy_complex2real (Vector)

  MQC_Vector_Copy_Complex2Real is a subroutine that copies a complex MQC_Vector into its real vector

- type(mqc_scalar) function mqc_vector_scalar_at (Vec, I)

  MQC_Vector_Scalar_At is a function that returns the ith element of a MQC vector as an MQC scalar

- type(mqc_vector) function mqc_vector_vector_at (Vec, I, J)

  MQC_Vector_Vector_At is a function that returns the vector at the specified subvector of MQC_Vector

- subroutine mqc_set_vector2integerarray (ArrayOut, VectorIn)

  MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array

- subroutine mqc_set_vector2realarray (ArrayOut, VectorIn)

MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array

- subroutine mqc_set_vector2complexarray (ArrayOut, VectorIn)

  MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array
- subroutine mqc_set_array2vector_integer (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector
- subroutine mqc_set_array2vector_real (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector
- subroutine mqc_set_array2vector_complex (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector
- subroutine mqc_set_vector2vector (VectorOut, VectorIn)

  MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector
- type(mqc_vector) function mqc_vectorvectorsum (Vector1In, Vector2In)

  MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_vectorvectordifference (Vector1In, Vector2In)

  MQC_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_scalarvectorsum (ScalarIn, VectorIn)

  MQC_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector
- type(mqc_vector) function mqc_scalarvectordifference (ScalarIn, VectorIn)

  MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector
- type(mqc_vector) function mqc_elementvectorproduct (Vector1In, Vector2In)

  MQC_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector
- type(mqc_vector) function mqc_vector_transpose (Vector)

  MQC_Vector_Transpose is a function that returns the transpose of an MQC vector
- type(mqc_vector) function mqc_vector_conjugate_transpose (Vector)

  MQC_Vector_Conjugate_Transpose is a function that returns the conjugate transpose of an MQC vector
- type(mqc_scalar) function mqc_vectorvectordotproduct (Vector1, Vector2)

  MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors
- type(mqc_matrix) function mqc_outer (VA, VB)

  MQC_Outer is a function that returns the outer product of two MQC vectors
- type(mqc_vector) function mqc_crossproduct (Vector1In, Vector2In)

  MQC_CrossProduct is a function that returns the cross product of two MQC vectors
- subroutine mqc_print_vector_algebra1 (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)

  MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector
- type(mqc_vector) function mqc_vector_cast_integer (VA)

  MQC_vector_cast_integer is a function that converts an MQC vector to its integer space
- type(mqc_vector) function mqc_vector_cast_real (VA)

  MQC_vector_cast_real is a function that converts an MQC vector to its real space
- type(mqc_vector) function mqc_vector_cast_complex (VA)

  MQC_vector_cast_complex is a function that converts an MQC vector to its complex space
- subroutine mqc_vector_scalar_put (Vector, Scalar, I)

  MQC_Vector_Scalar_Put is a subroutine that updates the value of the ith element of a MQC vector with the value of a MQC scalar
- subroutine mqc_vector_scalar_increment (Vector, Scalar, I)

  MQC_Vector_Scalar_Increment is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar
- subroutine mqc_vector_vector_put (Vector, VectorIn, I)

MQC_Vector_Vector_Put is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector

- subroutine mqc_vector_initialize (Vector, Length, Scalar)

  MQC_Vector_Initialize is a subroutine that initializes a MQC vector
- type(mqc_vector) function mqc_scalarvectorproduct (Scalar, Vector)

  MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector
- type(mqc_vector) function mqc_vectorscalarproduct (vector, scalar)

  MQC_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar
- type(mqc_vector) function mqc_vectorscalardivide (vector, scalar)

  MQC_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar
- type(mqc_vector) function mqc_realvectorproduct (RealIn, Vector)

  MQC_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector
- type(mqc_vector) function mqc_vectorrealproduct (vector, realIn)

  MQC_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar
- type(mqc_vector) function mqc_vectorrealdivide (vector, realIn)

  MQC_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer
- type(mqc_vector) function mqc_integervectorproduct (intIn, Vector)

  MQC_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector
- type(mqc_vector) function mqc_vectorintegerproduct (vector, intIn)

  MQC_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar
- type(mqc_vector) function mqc_vectorintegerdivide (vector, intIn)

  MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar
- type(mqc_vector) function mqc_complexvectorproduct (CompIn, Vector)

  MQC_ComplexVectorProduct is a function that returns the product of an intrinsic complex scalar and a MQC vector
- type(mqc_vector) function mqc_vectorcomplexproduct (vector, compIn)

  MQC_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar
- type(mqc_vector) function mqc_vectorcomplexdivide (vector, compIn)

  MQC_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar
- type(mqc_scalar) function mqc_vector_norm (vector, methodIn)

  MQC_Vector_Norm is a function that returns the norm of an MQC vector
- logical function mqc_vector_isallocated (Vector)

  MQC_Vector_isAllocated is a function that returns TRUE is an MQC vector is allocated and FALSE if it is not
- subroutine mqc_vector_push (Vector, Scalar)

  MQC_Vector_Push is a function that adds a value to the end of a MQC vector
- subroutine mqc_vector_unshift (Vector, Scalar)

  MQC_Vector_Unshift is a function that adds a value to the beginning of a MQC vector
- type(mqc_scalar) function mqc_vector_pop (Vector)

  MQC_Vector_Pop is a function that removes a value from the end of a MQC vector and returns it
- type(mqc_scalar) function mqc_vector_shift (Vector)

  MQC_Vector_Shift is a function that removes a value from the beginning of a MQC vector and returns it
- type(mqc_scalar) function mqc_vector_maxval (Vector)

  MQC_Vector_MaxVal is a function that returns the largest value in an MQC vector

- type(mqc_scalar) function mqc_vector_minval (Vector)

    MQC_Vector_MinVal is a function that returns the smallest value in an MQC vector
- integer function mqc_vector_maxloc (Vector)

    MQC_Vector_MaxLoc is a function that returns the index of the largest value in an MQC vector
- integer function mqc_vector_minloc (Vector)

    MQC_Vector_MinLoc is a function that returns the index of the smallest value in an MQC vector
- type(mqc_vector) function mqc_vector_argsort (Vector)

    MQC_Vector_Argsort is a function that returns the indices of an an MQC vector sorted from low to high
- subroutine mqc_vector_sort (Vector, idx)

    MQC_Vector_Sort is a function that returns an MQC vector sorted from low to high unless optional index order is present
- subroutine mqc_vector_sqrt (A)

    MQC_Vector_Sqrt is a function that returns the square root of all elements of an MQC vector
- type(mqc_vector) function mqc_vector_abs (A)

    MQC_Vector_Abs is a function that returns the absolute value of all elements of an MQC vector
- subroutine mqc_vector_power (A, P)

    MQC_Vector_Power is a function that returns the value of all elements of an MQC vector raised to a power
- type(mqc_vector) function mqc_vector_complex_realpart (A)

    MQC_Vector_Complex_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector
- type(mqc_vector) function mqc_vector_complex_imagpart (A)

    MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector
- type(mqc_vector) function mqc_vector_cmplx (Vector1, Vector2)

    MQC_Vector_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector
- subroutine mqc_matrix_diagonalize (A, EVals, EVecs)

    MQC_Matrix_Diagonalize is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors
- type(mqc_matrix) function mqc_matrix_cast_real (MA)

    MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real space
- type(mqc_matrix) function mqc_matrix_cast_integer (MA)

    MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its integer space
- type(mqc_matrix) function mqc_matrix_cast_complex (MA)

    MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its complex space
- type(mqc_scalar) function mqc_matrix_scalar_at (Mat, I, J)

    MQC_Matrix_Scalar_At is a function that returns the value of an element of a MQC matrix
- type(mqc_vector) function mqc_matrix_vector_at (Mat, Rows, Cols)

    MQC_Matrix_Vector_At is a function that returns the subvector of an MQC matrix
- recursive subroutine mqc_matrix_vector_put (Mat, VectorIn, Rows, Cols)

    MQC_Matrix_Vector_Put is a subroutine that writes a subvector to the specified position of a MQC matrix
- type(mqc_matrix) function mqc_matrix_matrix_at (Mat, Rows, Cols)

    MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix
- subroutine mqc_matrix_diagmatrix_put_vector (diagVectorIn, mat)

    MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

- subroutine mqc_matrix_diagmatrix_put_integer (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_matrix_diagmatrix_put_real (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_matrix_diagmatrix_put_complex (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector
- subroutine mqc_matrix_symmmatrix_put_integer (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_matrix_symmmatrix_put_real (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_matrix_symmmatrix_put_complex (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector
- recursive subroutine mqc_matrix_matrix_put (Mat, MatrixIn, Rows, Cols)

    MQC_Matrix_Matrix_Put is a subroutine that writes a submatrix to the specified position of a MQC matrix
- integer(kind=int64) function symindexhash (i, j, k, l)

    SymIndexHash is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor
- type(mqc_matrix) function mqc_elementmatrixproduct (A, B)

    MQC_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices
- type(mqc_matrix) function mqc_elementmatrixdivide (A, B)

    MQC_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices
- logical function mqc_matrix_test_symmetric (Matrix, Option)

    MQC_Matrix_Test_Symmetric is a function that tests a MQC matrix for symmetry
- logical function mqc_matrix_test_diagonal (Matrix)

    MQC_Matrix_Test_Diagonal is a function that tests a MQC matrix to determine if it is diagonal
- subroutine mqc_allocate_matrix (M, N, Matrix, Data_Type, Storage)

    MQC_Allocate_Matrix is used to allocate a matrix type variable of the MQC_Matrix class
- subroutine mqc_deallocate_matrix (Matrix)

    MQC_Deallocate_Matrix is used to deallocate a matrix type variable of the MQC_Matrix class
- logical function mqc_matrix_isallocated (Matrix)

    MQC_Matrix_isAllocate is a function that returns the allocation status of a MQC_Matrix variable
- subroutine mqc_set_integerarray2matrix (MatrixOut, ArrayIn)

    MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array
- subroutine mqc_set_realarray2matrix (MatrixOut, ArrayIn)

    MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array
- subroutine mqc_set_complexarray2matrix (MatrixOut, ArrayIn)

    MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array
- subroutine mqc_set_matrix2integerarray (ArrayOut, MatrixIn)

    MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix
- subroutine mqc_set_matrix2realarray (ArrayOut, MatrixIn)

    MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix
- subroutine mqc_set_matrix2complexarray (ArrayOut, MatrixIn)

MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix

- subroutine mqc_set_matrix2matrix (MatrixOut, MatrixIn)

  MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix
- subroutine mqc_print_matrix_algebra1 (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)

  MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix
- subroutine mqc_matrix_copy_int2real (Matrix)

  MQC_Matrix_Copy_Int2Real is a subroutine used to copy an integer MQC matrix into its real space
- subroutine mqc_matrix_copy_int2complex (Matrix)

  MQC_Matrix_Copy_Int2Complex is a subroutine used to copy an integer MQC matrix into its complex space
- subroutine mqc_matrix_copy_real2int (Matrix)

  MQC_Matrix_Copy_Real2Int is a subroutine used to copy a real MQC matrix into its integer space
- subroutine mqc_matrix_copy_real2complex (Matrix)

  MQC_Matrix_Copy_Real2Complex is a subroutine used to copy a real MQC matrix into its complex space
- subroutine mqc_matrix_copy_complex2int (Matrix)

  MQC_Matrix_Copy_Complex2Int is a subroutine used to copy a complex MQC matrix into its integer space
- subroutine mqc_matrix_copy_complex2real (Matrix)

  MQC_Matrix_Copy_Complex2Real is a subroutine used to copy a complex MQC matrix into its real space
- integer(kind=int64) function mqc_matrix_rows (Matrix)

  MQC_Matrix_Rows is a function used to return the number of rows of an MQC matrix
- integer(kind=int64) function mqc_matrix_columns (Matrix)

  MQC_Matrix_Columns is a function used to return the number of columns of an MQC matrix
- logical function mqc_matrix_havereal (Matrix)

  MQC_Matrix_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix
- logical function mqc_matrix_haveinteger (Matrix)

  MQC_Matrix_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix
- logical function mqc_matrix_havecomplex (Matrix)

  MQC_Matrix_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix
- logical function mqc_matrix_havefull (Matrix)

  MQC_Matrix_HaveFull is a function used to indicate if an MQC matrix is stored unpacked
- logical function mqc_matrix_havesymmetric (Matrix)

  MQC_Matrix_HaveSymmetric is a function used to indicate if an MQC matrix is stored symmetric-packed
- logical function mqc_matrix_havediagonal (Matrix)

  MQC_Matrix_HaveDiagonal is a function used to indicate if an MQC matrix is stored diagonal-packed
- type(mqc_matrix) function mqc_matrix_transpose (Matrix)

  MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix
- type(mqc_matrix) function mqc_matrix_conjugate_transpose (Matrix)

  MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate transpose of a MQC matrix
- type(mqc_matrix) function mqc_matrix_symmetrize (Matrix)

  MQC_Matrix_Symmetrize is a function that symmetrizes a MQC matrix
- subroutine mqc_matrix_full2symm (Matrix)

  MQC_Matrix_Full2Symm is a subroutine that converts an unpacked MQC matrix to symmetric-packed
- subroutine mqc_matrix_symm2full (Matrix, Option)

  MQC_Matrix_Symm2Full is a subroutine that converts a symmetry-packed MQC matrix to unpacked
- subroutine mqc_matrix_full2diag (Matrix)

  MQC_Matrix_Full2Diag is a subroutine that converts an unpacked MQC matrix to diagonal-packed
- subroutine mqc_matrix_diag2full (Matrix)

MQC_Matrix_Diag2Full is a subroutine that converts a diagonal-packed MQC matrix to unpacked
- subroutine mqc_matrix_symm2diag (Matrix)

  MQC_Matrix_Symm2Diag is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed
- subroutine mqc_matrix_diag2symm (Matrix)

  MQC_Matrix_Diag2Symm is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed
- type(mqc_matrix) function mqc_matrix_symm2full_func (Matrix)

  MQC_Matrix_Symm2Full_Func is a function that converts a symmetric- packed MQC matrix to unpacked
- subroutine matrix_symm2sq_integer (N, I_Symm, I_Sq)

  Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array
- subroutine matrix_symm2sq_real (N, A_Symm, A_Sq)

  Matrix_Symm2Sq_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array
- subroutine matrix_symm2sq_complex (N, A_Symm, A_Sq)

  Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array
- type(mqc_matrix) function mqc_vector2diagmatrix (vector)

  MQC_Vector2DiagMatrix is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector
- type(mqc_matrix) function mqc_matrixmatrixsum (MA, MB)

  MQC_MatrixMatrixSum is a function that sums two MQC matrices
- type(mqc_matrix) function mqc_matrixmatrixsubtract (MA, MB)

  MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices
- type(mqc_matrix) function mqc_matrixmatrixproduct (MA, MB)

  MQC_MatrixMatrixProduct is a function that computes the element- wise product of two MQC matrices
- type(mqc_matrix) function mqc_matrixmatrixdotproduct (MA, MB)
- type(mqc_vector) function mqc_matrixvectordotproduct (MA, VB)
- type(mqc_vector) function mqc_vectormatrixdotproduct (VA, MB)
- type(mqc_matrix) function mqc_matrixscalarproduct (Matrix, Scalar)
- type(mqc_matrix) function mqc_scalarmatrixproduct (Scalar, Matrix)
- type(mqc_scalar) function mqc_matrix_matrix_contraction (Matrix1, Matrix2)
- subroutine mqc_matrix_scalar_put (Matrix, Scalar, I, J)
- subroutine mqc_matrix_initialize (Matrix, Rows, Columns, Scalar, Storage)
- subroutine mqc_matrix_identity (matrix, n, m)
- subroutine mqc_matrix_set (matrix, scalar, storage)
- type(mqc_scalar) function mqc_matrix_norm (matrix, methodIn)
- type(mqc_scalar) function mqc_matrix_determinant (a)
- type(mqc_matrix) function mqc_matrix_inverse (a)
- type(mqc_scalar) function mqc_matrix_trace (matrix)
- subroutine mqc_matrix_generalized_eigensystem (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine mqc_matrix_svd (A, EVals, EUVecs, EVVecs)
- subroutine mqc_matrix_rms_max (A, rms_A, max_A)
- subroutine mqc_matrix_sqrt (A, eVals, eVecs)
- type(mqc_matrix) function mqc_givens_matrix (m_size, angle, p, q)
- subroutine mqc_allocate_r4tensor (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine mqc_deallocate_r4tensor (Tensor)
- type(mqc_scalar) function mqc_r4tensor_at (Tensor, I, J, K, L)
- subroutine mqc_r4tensor_put (Tensor, Element, I, J, K, L)
- subroutine mqc_print_r4tensor_algebra1 (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine mqc_set_array2tensor (TensorOut, ArrayIn)

- subroutine mqc_r4tensor_initialize (R4Tensor, I, J, K, L, Scalar)
- subroutine mqc_matrix_symmsymmr4tensor_put_real (r4Tensor, symmSymmMatrixIn)
- subroutine mqc_matrix_symmsymmr4tensor_put_complex (r4Tensor, symmSymmMatrixIn)
- logical function mqc_r4tensor_haveinteger (R4Tensor)
- logical function mqc_r4tensor_havereal (R4Tensor)
- logical function mqc_r4tensor_havecomplex (R4Tensor)

5.1.1 Detailed Description

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

Purpose:

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

1. MQC_Scalar:   Rank 0 array variable
2. MQC_Vector:   Rank 1 array variable
3. MQC_Matrix:   Rank 2 array variable
4. MQC_R4Tensor: Rank 3 array variable

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

Note that MQC_Algebra2 provides similar functionality with MQC_Array
objects that can dynamically adjust rank. However, the vast majority of
work can be performed using MQC_Algebra derived types which have been
more developed than MQC_Algebra2 derived types.
```

5.1.2 Function/Subroutine Documentation

5.1.2.1 bin_coeff()

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

Bin_Coeff returns the binomial coefficient of (n,k)

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.2 factorial()

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

Factorial returns the factorial of an integer

Purpose:

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

Parameters

in	*N*	`N is Integer(kind=int64)` `The argument of the factorial function.`

Author

L. M. Thompson

Date

2016

5.1.2.3 matrix_symm2sq_complex()

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

Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array

Purpose:

```
Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric-packed
intrinsic complex matrix to a rank-2 complex array.
TODO: Move this routine to MQC general
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.4 matrix_symm2sq_integer()

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

Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array

Purpose:

```
Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric-packed
intrinsic integer matrix to a rank-2 integer array.
TODO: Move this routine to MQC general
```

Parameters

in	*N*	`N is Integer(kind=int64)`
		`The leading dimension of symmetric-packed matrix I_Symm.`
		`unpacked.`

in	*I_Symm*	`I_Symm is Integer(kind=int64),Dimension(:)` `The symmetric-packed intrinsic integer matrix to be unpacked.`
out	*I_Sq*	`I_Sq is Integer(kind=int64),Dimension(N,N)` `The upacked intrinsic integer matrix output.`

Author

H. P. Hratchian

Date

2017

5.1.2.5 matrix_symm2sq_real()

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

Matrix_Symm2Sq_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array

Purpose:

```
Matrix_Symm2Sq_Real is a subroutine that converts a symmetric-packed
intrinsic real matrix to a rank-2 real array.
TODO: Move this routine to MQC general
```

Parameters

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

Author

H. P. Hratchian

Date

2017

5.1.2.6 mqc_allocate_matrix()

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

MQC_Allocate_Matrix is used to allocate a matrix type variable of the MQC_Matrix class

Purpose:

```
MQC_Allocate_Matrix is a subroutine used to allocate a matrix type variable
of the MQC_Matrix class. The following options are available:

1. Data_Type = 'Real' declares the MQC_Matrix variable to be of real type.
2. Data_Type = 'Integer' declares the MQC_Matrix variable to be of integer type.
3. Data_Type = 'Complex' declares the MQC_Matrix variable to be of complex type.
1. Data_Type = 'StorFull' declares the MQC_Matrix variable to be unpacked.
2. Data_Type = 'StorSymm' declares the MQC_Matrix variable to be symmetric packed.
3. Data_Type = 'StorDiag' declares the MQC_Matrix variable to be diagonal packed.
```

Parameters

`in`	*M*	`M is Integer(kind=int64)` `M is the number of rows of Matrix.`
`in`	*N*	`N is Integer(kind=int64)` `N is the number of columns of Matrix.`
`in,out`	*Matrix*	`Matrix is Class(MQC_Matrix)` `The MQC matrix to be allocated.`

`in`	*Data_Type*	`Data_Type is Character(Len=*)`
		`= 'Real':    the MQC_Scalar is real`
		`= 'Integer': the MQC_Scalar is integer`
		`= 'Complex': the MQC_Scalar is complex.`

`in`	*Storage*	`Storage is Character(Len=*)`
		`= 'StorFull': the MQC_Scalar is unpacked`
		`= 'StorSymm': the MQC_Scalar is symmetric packed`
		`= 'StorDiag': the MQC_Scalar is diagonal packed.`

Author

H. P. Hratchian

L. M. Thompson

Date

2016

5.1.2.7 mqc_allocate_r4tensor()

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

5.1.2.8 mqc_allocate_scalar()

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

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

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.9 mqc_allocate_vector()

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

MQC_Allocate_Vector is used to allocate a vector type variable of the MQC_Vector class

Purpose:

```
MQC_Allocate_Vector is a subroutine used to allocate a vector type variable
of the MQC_Vector class. The following options are available:

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

Parameters

in	*N*	`N is Integer(kind=int64)` `The length of the MQC_Vector variable`
in,out	*Vector*	`Vector is Type(MQC_Vector)` `The name of the MQC_Vector variable`

in	*Data_Type*	`Data_Type is Character(Len=*)`
		`= 'Real':    the MQC_Vector is real`
		`= 'Integer': the MQC_Vector is integer`
		`= 'Complex': the MQC_Vector is complex`

Author

H. P. Hratchian

Date

2016

5.1.2.10 mqc_complexscalaradd()

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

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

Purpose:

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

Parameters

in	*Complex↔ In*	Complex is Complex(kind=real64) The intrinsic complex variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum.

Author

L. M. Thompson

Date

2019

5.1.2.11 mqc_complexscalardivide()

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

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

Purpose:

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

Parameters

in	*Complex↔ In*	ComplexIn is Complex(kind=real64) The intrinsic complex variable numerator.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator.

Author

L. M. Thompson

Date

2019

5.1.2.12 mqc_complexscalarmultiply()

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

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

Purpose:

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

Parameters

in	*ComplexIn*	Complex is Complex(kind=real64) The intrinsic complex variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply.

Author

L. M. Thompson

Date

2019

5.1.2.13 mqc_complexscalarsubtract()

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

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

Purpose:

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

Parameters

in	*Complex↩ In*	`ComplexIn is Complex(kind=real64)` `The intrinsic complex to subtract from.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract.`

Author

L. M. Thompson

Date

2019

5.1.2.14 mqc_complexvectorproduct()

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

MQC_ComplexVectorProduct is a function that returns the product of an intrinsic complex scalar and a MQC vector

Purpose:

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

Parameters

in	*Comp↩ In*	`CompIn is Complex(kind=real64)` `The intrinsic complex to multiply.`
in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.15 mqc_crossproduct()

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

MQC_CrossProduct is a function that returns the cross product of two MQC vectors

Purpose:

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

Parameters

in	*Vector1In*	Vector1In is Type(MQC_Vector) The first MQC vector.
in	*Vector2In*	Vector2In is Type(MQC_Vector) The second MQC vector.

Author

L. M. Thompson

Date

2016

5.1.2.16 mqc_deallocate_matrix()

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

MQC_Deallocate_Matrix is used to deallocate a matrix type variable of the MQC_Matrix class

Purpose:

```
MQC_Deallocate_Matrix is a subroutine used to deallocate a matrix type variable
of the MQC_Matrix class.
```

Parameters

in,out	*Matrix*	Matrix is Class(MQC_Matrix) The MQC matrix to be deallocated.

Author

L. M. Thompson

Date

2016

5.1.2.17 mqc_deallocate_r4tensor()

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

5.1.2.18 mqc_deallocate_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.19 mqc_deallocate_vector()

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

MQC_Deallocate_Vector is used to deallocate a vector type variable of the MQC_Vector class

Purpose:

```
MQC_Deallocate_Vector is a subroutine used to deallocate a vector type variable
of the MQC_Vector class.
```

Parameters

in,out	*Vector*	`Vector is Type(MQC_Vector)` `The name of the MQC_Vector variable to deallocate.`

Author

H. P. Hratchian

Date

2016

5.1.2.20 mqc_elementmatrixdivide()

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

MQC_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices

Purpose:

```
MQC_ElementMatrixDivide is a function that returns the element-wise quotient
of two MQC matrices.
```

Parameters

in	*A*	A is type(mqc_matrix) The matrix with elements being the numerator.
in	*B*	B is type(mqc_matrix) The matrix with elements being the denominator.

Author

X. Sheng

Date

2017

5.1.2.21 mqc_elementmatrixproduct()

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

MQC_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices

Purpose:

```
MQC_ElementMatrixProduct is a function that returns the element-wise product
of two MQC matrices.
```

Parameters

in	*A*	`A is type(mqc_matrix)` `The first matrix to element-wise multiply.`
in	*B*	`B is type(mqc_matrix)` `The second matrix to element-wise multiply.`

Author

X. Sheng

Date

2017

5.1.2.22 mqc_elementvectorproduct()

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

MQC_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector

Purpose:

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

Parameters

in	*Vector1↩ In*	`Vector1In is Type(MQC_Vector)` `The frist MQC vector to multiply elementwise.`
in	*Vector2↩ In*	`Vector2In is Type(MQC_Vector)` `The second MQC vector to multiply elementwise.`

Author

L. M. Thompson

Date

2016

5.1.2.23 mqc_givens_matrix()

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

5.1.2.24 mqc_input_complex_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.25 mqc_input_integer_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Type(MQC_Scalar) The name of the output variable.
in	*ScalarIn*	ScalarIn is Integer(kind=int64) The value of the input variable.

Author

L. M. Thompson

Date

2016

5.1.2.26 mqc_input_real_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	`ScalarOut is Type(MQC_Scalar)` `The name of the output variable.`
in	*ScalarIn*	`ScalarIn is Real(kind=real64)` `The value of the input variable.`

Author

L. M. Thompson

Date

2016

5.1.2.27 mqc_integergtscalar()

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

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

Purpose:

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

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

Parameters

in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer that will be tested.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2019

5.1.2.28 mqc_integerlescalar()

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

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

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.29 mqc_integerscalaradd()

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

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

Purpose:

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

Parameters

in	*IntegerIn*	IntegerIn is Integer(kind=int64) The intrinsic integer variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum.

Author

L. M. Thompson

Date

2019

5.1.2.30 mqc_integerscalardivide()

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

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

Purpose:

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

Parameters

in	*Integer↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable numerator.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable denominator.`

Author

L. M. Thompson

Date

2019

5.1.2.31 mqc_integerscalarmultiply()

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

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

Purpose:

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

Parameters

in	*Integer↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable to multiply.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.32 mqc_integerscalarsubtract()

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

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

Purpose:

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

Parameters

in	*IntegerIn*	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract from.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract.

Author

L. M. Thompson

Date

2019

5.1.2.33 mqc_integervectorproduct()

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

MQC_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector

Purpose:

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

Parameters

in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer to multiply.`
in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.34 mqc_length_vector()

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

MQC_Length_Vector is used to return the length of an MQC vector

Purpose:

```
MQC_Length_Vector is used to return the length of an MQC vector. If the vector
vector is NOT allocated, the length is returned as 0.
```

Parameters

in,out	*Vector*	`Vector is Type(MQC_Vector)` `The name of the MQC_Vector variable whose length will be returned.`

Author

H. P. Hratchian

Date

2016

5.1.2.35 mqc_matrix_cast_complex()

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

MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its complex space

Purpose:

```
MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its
complex space.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.36 mqc_matrix_cast_integer()

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

MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its integer space

Purpose:

```
MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its
integer space.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.37 mqc_matrix_cast_real()

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

MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real space

Purpose:

```
MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real
space.
```

Parameters

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

Author

X. Sheng

Date

2017

5.1.2.38 mqc_matrix_columns()

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

MQC_Matrix_Columns is a function used to return the number of columns of an MQC matrix

Purpose:

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

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix)` `The MQC matrix to be tested.`

Author

L. M. Thompson

Date

2016

5.1.2.39 mqc_matrix_conjugate_transpose()

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

MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate transpose of a MQC matrix

Purpose:

```
MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate
transpose of a MQC matrix.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix)` `The MQC matrix to be conjugate transposed.`

Author

L. M. Thompson

Date

2016

5.1.2.40 mqc_matrix_copy_complex2int()

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

MQC_Matrix_Copy_Complex2Int is a subroutine used to copy a complex MQC matrix into its integer space

Purpose:

```
MQC_Matrix_Copy_Complex2Int is a subroutine used to copy a complex MQC matrix
matrix into its integer space.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to integer.

Author

L. M. Thompson

Date

2017

5.1.2.41 mqc_matrix_copy_complex2real()

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

MQC_Matrix_Copy_Complex2Real is a subroutine used to copy a complex MQC matrix into its real space

Purpose:

```
MQC_Matrix_Copy_Complex2Real is a subroutine used to copy a complex MQC matrix
matrix into its real space.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to real.

Author

L. M. Thompson

Date

2017

5.1.2.42 mqc_matrix_copy_int2complex()

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

MQC_Matrix_Copy_Int2Complex is a subroutine used to copy an integer MQC matrix into its complex space

Purpose:

```
MQC_Matrix_Copy_Int2Complex is a subroutine used to copy an integer MQC matrix
into its complex space.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to complex.

Author

L. M. Thompson

Date

2017

5.1.2.43 mqc_matrix_copy_int2real()

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

MQC_Matrix_Copy_Int2Real is a subroutine used to copy an integer MQC matrix into its real space

Purpose:

```
MQC_Matrix_Copy_Int2Real is a subroutine used to copy an integer MQC matrix
into its real space.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix)` `The MQC matrix to be converted to real.`

Author

L. M. Thompson

Date

2016

5.1.2.44 mqc_matrix_copy_real2complex()

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

MQC_Matrix_Copy_Real2Complex is a subroutine used to copy a real MQC matrix into its complex space

Purpose:

```
MQC_Matrix_Copy_Real2Complex is a subroutine used to copy a real MQC matrix
matrix into its complex space.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix)` `The MQC matrix to be converted to complex.`

Author

L. M. Thompson

Date

2017

5.1.2.45 mqc_matrix_copy_real2int()

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

MQC_Matrix_Copy_Real2Int is a subroutine used to copy a real MQC matrix into its integer space

Purpose:

```
MQC_Matrix_Copy_Real2Int is a subroutine used to copy a real MQC matrix matrix
into its integer space.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be converted to integer.

Author

L. M. Thompson

Date

2016

5.1.2.46 mqc_matrix_determinant()

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

5.1.2.47 mqc_matrix_diag2full()

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

MQC_Matrix_Diag2Full is a subroutine that converts a diagonal-packed MQC matrix to unpacked

Purpose:

```
MQC_Matrix_Diag2Full is a subroutine that converts a diagonal-packed MQC matrix
to unpacked.
  TODO: make tests for diagonal structure more efficient.
```

Parameters

in,out	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.

Author

L. M. Thompson

Date

2017

5.1.2.48 mqc_matrix_diag2symm()

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

MQC_Matrix_Diag2Symm is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed

Purpose:

```
MQC_Matrix_Diag2Symm is a subroutine that converts a diagonal-packed MQC matrix
to symmetry-packed.
  TODO: make tests for diagonal structure more efficient.
```

Parameters

in,out	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.

Author

L. M. Thompson

Date

2017

5.1.2.49 mqc_matrix_diagmatrix_put_complex()

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

MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic complex vector.
```

Parameters

in,out	*mat*	`Mat is class(MQC_Matrix)` `MQC matrix to overwrite with output diagonal matrix.`
in	*Diag↩ MatrixIn*	`DiagMatrixIn is complex(kind=real64),dimension(:)` `Intrinsic complex vector to write as diagonal matrix.`

Author

L. M. Thompson

Date

2017

5.1.2.50 mqc_matrix_diagmatrix_put_integer()

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

MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic integer vector.
```

Parameters

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

Author

H. P. Hratchian

Date

2017

5.1.2.51 mqc_matrix_diagmatrix_put_real()

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

MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic real vector.
```

Parameters

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

Author

H. P. Hratchian

Date

2017

5.1.2.52 mqc_matrix_diagmatrix_put_vector()

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

MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC
matrix with elements defined by values in a MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2018

5.1.2.53 mqc_matrix_diagonalize()

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

MQC_Matrix_Diagonalize is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors

Purpose:

```
MQC_Matrix_Diagonalize is a subroutine that takes a symmetric or hermitian MQC
matrix and optionally returns eigenvalues to a MQC vector and/or eigenvectors
to a MQC matrix.
```

Parameters

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

Author

X. Sheng

L. M. Thompson

Date

2017

5.1.2.54 mqc_matrix_full2diag()

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

MQC_Matrix_Full2Diag is a subroutine that converts an unpacked MQC matrix to diagonal-packed

Purpose:

```
MQC_Matrix_Full2Diag is a subroutine that converts an unpacked MQC matrix to
diagonal-packed.
  TODO: make tests for diagonal structure more efficient.
```

Parameters

in,out	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be diagonal packed.

Author

L. M. Thompson

Date

2016

5.1.2.55 mqc_matrix_full2symm()

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

MQC_Matrix_Full2Symm is a subroutine that converts an unpacked MQC matrix to symmetric-packed

Purpose:

```
MQC_Matrix_Full2Symm is a subroutine that converts an unpacked MQC matrix to
symmetric-packed.
  TODO: make tests for symmetry more efficient.
```

Parameters

in,out	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be symmetric packed.

Author

L. M. Thompson

Date

2016

5.1.2.56 mqc_matrix_generalized_eigensystem()

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

5.1.2.57 mqc_matrix_havecomplex()

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

MQC_Matrix_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix

Purpose:

```
MQC_Matrix_HaveComplex is a function that returns TRUE if an MQC matrix has an
allocated complex matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.58 mqc_matrix_havediagonal()

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

MQC_Matrix_HaveDiagonal is a function used to indicate if an MQC matrix is stored diagonal-packed

Purpose:

```
MQC_Matrix_HaveDiagonal is a function that returns TRUE if an MQC matrix is stored
diagonal-packed and FALSE if it is not.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix) The MQC matrix to be tested.`

Author

L. M. Thompson

Date

2017

5.1.2.59 mqc_matrix_havefull()

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

MQC_Matrix_HaveFull is a function used to indicate if an MQC matrix is stored unpacked

Purpose:

```
MQC_Matrix_HaveFull is a function that returns TRUE if an MQC matrix is stored
unpacked and FALSE if it is not.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix) The MQC matrix to be tested.`

Author

L. M. Thompson

Date

2017

5.1.2.60 mqc_matrix_haveinteger()

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

MQC_Matrix_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix

Purpose:

```
MQC_Matrix_HaveInteger is a function that returns TRUE if an MQC matrix has an
allocated integer matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2016

5.1.2.61 mqc_matrix_havereal()

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

MQC_Matrix_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix

Purpose:

```
MQC_Matrix_HaveReal is a function that returns TRUE if an MQC matrix has an
allocated real matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2016

5.1.2.62 mqc_matrix_havesymmetric()

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

MQC_Matrix_HaveSymmetric is a function used to indicate if an MQC matrix is stored symmetric-packed

Purpose:

```
MQC_Matrix_HaveSymmetric is a function that returns TRUE if an MQC matrix is stored
symmetric-packed and FALSE if it is not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.63 mqc_matrix_identity()

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

5.1.2.64 mqc_matrix_initialize()

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

5.1.2.65 mqc_matrix_inverse()

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

5.1.2.66 mqc_matrix_isallocated()

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

MQC_Matrix_isAllocate is a function that returns the allocation status of a MQC_Matrix variable

Purpose:

```
MQC_Matrix_isAllocate is a function that returns the allocation status of a
MQC_Matrix variable. The function returns TRUE if the matrix is allocated
and FALSE if the matrix is not allocated.
```

Parameters

in,out	*Matrix*	Matrix is Class(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.67 mqc_matrix_matrix_at()

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

MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix

Purpose:

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

Parameters

in	*Mat*	Mat is Class(MQC_Matrix) Name of the input matrix variable

in	*rows*	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	*Cols*	Cols is Integer(kind=int64),Dimension(:)
		If = [A,B]: output is submatrix of columns A to B
		  If (A,B)>0 column count is from first index
		  If (A,B)<0 column count is from last index
		If = [0]: submatrix of columns equivalent to [1,-1].

Author

L. M. Thompson

Date

2017

5.1.2.68 mqc_matrix_matrix_contraction()

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

5.1.2.69 mqc_matrix_matrix_put()

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

MQC_Matrix_Matrix_Put is a subroutine that writes a submatrix to the specified position of a MQC matrix

Purpose:

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

Parameters

in,out	*Mat*	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.
in	*Matrix↵ In*	MatrixIn is Type(MQC_Matrix) The submatrix to overwrite at the specified elements of Mat.

in	*Rows*	Rows is Integer(kind=int64),Dimension(:)
		The specification of the rows to include in the subvector.
		If = [A,B]: output is subvector of rows A to B
		  If (A,B)>0 row count is from first index
		  If (A,B)<0 row count is from last index
		If = [0]: subvector of rows equivalent to [1,-1].

in	*Cols*	Cols is Integer(kind=int64),Dimension(:)
		The specification of the columns to include in the subvector.
		If = [A,B]: output is subvector of columns A to B
		  If (A,B)>0 column count is from first index
		  If (A,B)<0 column count is from last index
		If = [0]: subvector of columns equivalent to [1,-1].

Author

L. M. Thompson

Date

2017

5.1.2.70 mqc_matrix_norm()

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

5.1.2.71 mqc_matrix_rms_max()

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

5.1.2.72 mqc_matrix_rows()

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

MQC_Matrix_Rows is a function used to return the number of rows of an MQC matrix

Purpose:

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

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2016

5.1.2.73 mqc_matrix_scalar_at()

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

MQC_Matrix_Scalar_At is a function that returns the value of an element of a MQC matrix

Purpose:

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

Parameters

in	*Mat*	Mat is Class(MQC_Matrix) The MQC matrix to return the value of the (I,J)th element.

in	*I*	I is Integer(kind=int64)
		The row of the element in MQC matrix.
		  If I>0 row count is from first index
		  If I<0 row count is from last index.

in	*J*	J is Integer(kind=int64)
		The column of the element in MQC matrix.
		  If J>0 row count is from first index
		  If J<0 row count is from last index.

Author

X. Sheng

L. M. Thompson

Date

2017

5.1.2.74 mqc_matrix_scalar_put()

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

5.1.2.75 mqc_matrix_set()

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

5.1.2.76 mqc_matrix_sqrt()

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

5.1.2.77 mqc_matrix_svd()

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

5.1.2.78 mqc_matrix_symm2diag()

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

MQC_Matrix_Symm2Diag is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed

Purpose:

```
MQC_Matrix_Symm2Diag is a subroutine that converts a symmetry-packed MQC matrix to
diagonal-packed.
  TODO: make tests for diagonal structure more efficient.
```

Parameters

in,out	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be diagonal packed.

Author

L. M. Thompson

Date

2017

5.1.2.79 mqc_matrix_symm2full()

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

MQC_Matrix_Symm2Full is a subroutine that converts a symmetry-packed MQC matrix to unpacked

Purpose:

```
MQC_Matrix_Symm2Full is a subroutine that converts a symmetry-packed MQC matrix
to unpacked. The following options are available:
  Option = 'symmetric' unpacks as if matrix is symmetric.
  Option = 'antisymmetric' unpacks as if matrix is antisymmetric.
  Option = 'hermitian' unpacks as if matrix is hermitian.
  Option = 'antihermitian' unpacks as if matrix is antihermitian.

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

Parameters

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

Author

L. M. Thompson

Date

2016, 2018

5.1.2.80 mqc_matrix_symm2full_func()

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

MQC_Matrix_Symm2Full_Func is a function that converts a symmetric- packed MQC matrix to unpacked

Purpose:

```
MQC_Matrix_Symm2Full_Func is a function that converts a symmetric-packed MQC matrix
to unpacked.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be unpacked.

Author

X. Sheng

Date

2017

5.1.2.81 mqc_matrix_symmetrize()

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

MQC_Matrix_Symmetrize is a function that symmetrizes a MQC matrix

Purpose:

```
MQC_Matrix_Symmetrize is a function that symmetrizes a MQC matrix.
  TODO: options to antisymmetrize, hermitianize and antihermitianize.
```

Parameters

in	*Matrix*	`Matrix is Type(MQC_Matrix)` `The MQC matrix to be symmetrized.`

Author

L. M. Thompson

Date

2016

5.1.2.82 mqc_matrix_symmmatrix_put_complex()

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

MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic complex
vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.83 mqc_matrix_symmmatrix_put_integer()

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

MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic integer
vector.
```

Parameters

in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	*Symm← MatrixIn*	SymmMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as symmetric-packed matrix.

Author

H. P. Hratchian

Date

2017

5.1.2.84 mqc_matrix_symmmatrix_put_real()

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

MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic real
vector.
```

Parameters

in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	*Symm← MatrixIn*	SymmMatrixIn is real(kind=real64),dimension(:) Intrinsic real vector to write as symmetric-packed matrix.

Author

H. P. Hratchian

Date

2017

5.1.2.85 mqc_matrix_symmsymmr4tensor_put_complex()

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

5.1.2.86 mqc_matrix_symmsymmr4tensor_put_real()

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

5.1.2.87 mqc_matrix_test_diagonal()

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

MQC_Matrix_Test_Diagonal is a function that tests a MQC matrix to determine if it is diagonal

Purpose:

```
MQC_Matrix_Test_Diagonal is a function that tests a MQC matrix to determine
if it is diagonal. The function returns TRUE if the matrix is diagonal and
FALSE if the matrix is not diagonal.
```

Parameters

in	*Matrix*	Matrix is Class(mqc_matrix) The matrix to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.88 mqc_matrix_test_symmetric()

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

MQC_Matrix_Test_Symmetric is a function that tests a MQC matrix for symmetry

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.89 mqc_matrix_trace()

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

5.1.2.90 mqc_matrix_transpose()

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

MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix

Purpose:

```
MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be transposed.

Author

L. M. Thompson

X. Sheng

Date

2016, 2017

5.1.2.91 mqc_matrix_vector_at()

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

MQC_Matrix_Vector_At is a function that returns the subvector of an MQC matrix

Purpose:

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

Parameters

in	*Mat*	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.

in	*Rows*	Rows is Integer(kind=int64),Dimension(:)
		The specification of the rows to include in the
		subvector.
		If = [A,B]: output is subvector of rows A to B
		  If (A,B)>0 row count is from first index
		  If (A,B)<0 row count is from last index
		If = [0]: subvector of rows equivalent to [1,-1].

in	*Cols*	Cols is Integer(kind=int64),Dimension(:)
		The specification of the columns to include in the
		subvector.
		If = [A,B]: output is subvector of columns A to B
		  If (A,B)>0 column count is from first index
		  If (A,B)<0 column count is from last index
		If = [0]: subvector of columns equivalent to [1,-1].

Author

L. M. Thompson

Date

2017

5.1.2.92 mqc_matrix_vector_put()

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

MQC_Matrix_Vector_Put is a subroutine that writes a subvector to the specified position of a MQC matrix

Purpose:

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

Parameters

in,out	*Mat*	Mat is Class(MQC_Matrix) The MQC matrix from which to return the subvector.
in	*Vector↩ In*	VectorIn is Type(MQC_Vector) The subvector to overwrite at the specified elements of Mat.

in	*Rows*	Rows is Integer(kind=int64),Dimension(:)
		The specification of the rows to include in the subvector.
		If = [A,B]: output is subvector of rows A to B
		  If (A,B)>0 row count is from first index
		  If (A,B)<0 row count is from last index
		If = [0]: subvector of rows equivalent to [1,-1].

in	*Cols*	Cols is Integer(kind=int64),Dimension(:)
		The specification of the columns to include in the subvector.
		If = [A,B]: output is subvector of columns A to B
		  If (A,B)>0 column count is from first index
		  If (A,B)<0 column count is from last index
		If = [0]: subvector of columns equivalent to [1,-1].

Author

L. M. Thompson

Date

2017

5.1.2.93 mqc_matrixmatrixdotproduct()

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

5.1.2.94 mqc_matrixmatrixproduct()

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

MQC_MatrixMatrixProduct is a function that computes the element- wise product of two MQC matrices

Purpose:

```
MQC_MatrixMatrixProduct is a function that computes the element-wise product of
two MQC matrices.
```

Parameters

in	*MA*	MA is Type(MQC_Matrix) The first MQC matrix.
in	*MB*	MB is Type(MQC_Matrix) The second MQC matrix.

Author

H. P. Hratchian

Date

2017

5.1.2.95 mqc_matrixmatrixsubtract()

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

MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices

Purpose:

```
MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices.
```

Parameters

in	*MA*	MA is Type(MQC_Matrix) The matrix that MB will be subtracted from.
in	*MB*	MB is Type(MQC_Matrix) The matrix that will be subtracted from MA.

Author

H. P. Hratchian

Date

2017

5.1.2.96 mqc_matrixmatrixsum()

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

MQC_MatrixMatrixSum is a function that sums two MQC matrices

Purpose:

```
MQC_MatrixMatrixSum is a function that sums two MQC matrices.
```

Parameters

in	*MA*	MA is Type(MQC_Matrix) First MQC matrix to sum.
in	*MB*	MB is Type(MQC_Matrix) Second MQC matrix to sum.

Author

H. P. Hratchian

L. M. Thompson

Date

2017, 2018

5.1.2.97 mqc_matrixscalarproduct()

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

5.1.2.98 mqc_matrixvectordotproduct()

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

5.1.2.99 mqc_outer()

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

MQC_Outer is a function that returns the outer product of two MQC vectors

Purpose:

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

Parameters

in	*VA*	`VA is Type(MQC_Vector) The MQC column vector.`
in	*VB*	`VB is Type(MQC_Vector) The MQC row vector.`

Author

X. Sheng

Date

2017

5.1.2.100 mqc_output_complex_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	`ScalarOut is Complex(kind=real64) The name of the output variable.`
in	*ScalarIn*	`ScalarIn is Type(MQC_Scalar) The value of the input variable.`

Author

L. M. Thompson

Date

2017

5.1.2.101 mqc_output_integer_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Integer(kind=int64) The name of the output variable.
in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The value of the input variable.

Author

L. M. Thompson

Date

2016

5.1.2.102 mqc_output_mqcscalar_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	`ScalarOut is Type(MQC_Scalar)` `The name of the output variable.`
in	*ScalarIn*	`ScalarIn is Type(MQC_Scalar)` `The value of the input variable.`

Author

L. M. Thompson

Date

2016

5.1.2.103 mqc_output_real_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	`ScalarOut is Real(kind=real64)` `The name of the output variable.`
in	*ScalarIn*	`ScalarIn is Type(MQC_Scalar)` `The value of the input variable.`

Author

L. M. Thompson

Date

2016

5.1.2.104 mqc_print_matrix_algebra1()

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

MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix

Purpose:

```
MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix.
Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank
lines before or after output.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.105 mqc_print_r4tensor_algebra1()

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

5.1.2.106 mqc_print_scalar_algebra1()

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

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Class(MQC_Scalar)` `The variable to be printed.`
in	*IOut*	`IOut is Integer(kind=int64)` `The Fortran file number to print to.`
in	*Header*	`Header is Character(Len=*)` `The title to print along with Scalar.`

in	*Blank_At_Top*	`Blank_At_Top is Logical,Optional`
		`= .True.:  print blank line above output`
		`= .False.: do not print blank line above output.`

in	*Blank_At_Bottom*	`Blank_At_Bottom is Logical,Optional`
		`= .True.:  print blank line below output`
		`= .False.: do not print blank line below output.`

Author

L. M. Thompson

Date

2016

5.1.2.107 mqc_print_vector_algebra1()

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

MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector

Purpose:

```
MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector.
Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank
lines before or after output.
```

Parameters

in	*Vector*	Vector is Class(MQC_Vector) The variable to be printed.
in	*IOut*	IOut is Integer(kind=int64) The Fortran file number to print to.
in	*Header*	Header is Character(Len=*) The title to print along with Vector.
in	*Verbose*	Verbose is Logical,Optional Adds extra printing to output.
in	*Blank_At_Top*	Blank_At_Top is Logical,Optional = .True.: print blank line above output = .False.: do not print blank line above output.
in	*Blank_At_Bottom*	Blank_At_Bottom is Logical,Optional = .True.: print blank line below output = .False.: do not print blank line below output.

Author

L. M. Thompson

Date

2016

5.1.2.108 mqc_r4tensor_at()

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

5.1.2.109 mqc_r4tensor_havecomplex()

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

5.1.2.110 mqc_r4tensor_haveinteger()

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

5.1.2.111 mqc_r4tensor_havereal()

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

5.1.2.112 mqc_r4tensor_initialize()

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

5.1.2.113 mqc_r4tensor_put()

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

5.1.2.114 mqc_realgtscalar()

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

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

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.115 mqc_reallescalar()

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

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

Purpose:

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

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

Parameters

in	*Real↩ In*	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.2.116 mqc_realltscalar()

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

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

Purpose:

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

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

Parameters

in	*Real↔In*	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

5.1.2.117 mqc_realscalaradd()

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

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

Purpose:

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

Parameters

in	*realIn*	RealIn is Real(kind=real64) The intrinsic real variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to sum.

Author

L. M. Thompson

Date

2019

5.1.2.118 mqc_realscalardivide()

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

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

Purpose:

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

Parameters

in	*RealIn*	RealIn is Real(kind=real64) The intrinsic real variable numerator.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator.

Author

L. M. Thompson

Date

2019

5.1.2.119 mqc_realscalarmultiply()

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

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

Purpose:

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

Parameters

in	*realIn*	RealIn is Real(kind=real64) The intrinsic real variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply.

Author

L. M. Thompson

Date

2019

5.1.2.120 mqc_realscalarsubtract()

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

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

Purpose:

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

Parameters

in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real to subtract from.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract.`

Author

L. M. Thompson

Date

2019

5.1.2.121 mqc_realvectorproduct()

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

MQC_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector

Purpose:

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

Parameters

in	*Real↩ In*	`RealIn is Real(kind=real64)` `The real intrinsic to multiply.`
in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.122 mqc_scalar_acos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.123 mqc_scalar_asin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.124 mqc_scalar_atan()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.125 mqc_scalar_atan2()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.126 mqc_scalar_cmplx()

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

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

Purpose:

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

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The real part of MQC_Scalar_Cmplx.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The imaginary part of MQC_Scalar_Cmplx.

Author

L. M. Thompson

Date

2019

5.1.2.127 mqc_scalar_complex_conjugate()

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

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

Purpose:

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

Parameters

in	*Scalar↩In*	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.

Author

L. M. Thompson

Date

2018

5.1.2.128 mqc_scalar_complex_imagpart()

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

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

Purpose:

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

Parameters

in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.

Author

L. M. Thompson

Date

2019

5.1.2.129 mqc_scalar_complex_realpart()

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

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

Purpose:

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

Parameters

in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.

Author

L. M. Thompson

Date

2019

5.1.2.130 mqc_scalar_cos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.131 mqc_scalar_get_abs_value()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Class(MQC_Scalar)` `The MQC_Scalar to be tested.`

Author

A. Mahler

Date

2018

5.1.2.132 mqc_scalar_get_intrinsic_complex()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Class(MQC_Scalar)` `The MQC_Scalar to be tested.`

Author

L. M. Thompson

Date

2017

5.1.2.133 mqc_scalar_get_intrinsic_integer()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.134 mqc_scalar_get_intrinsic_real()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Class(MQC_Scalar) The MQC_Scalar to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.135 mqc_scalar_get_random_value()

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

MQC_Scalar_Get_Random_Value is a function that returns a random real value from a specified distribution

Purpose:

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

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

Parameters

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

Author

X. Dong

L. M. Thompson

Date

2019

5.1.2.136 mqc_scalar_havecomplex()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.137 mqc_scalar_haveinteger()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.138 mqc_scalar_havereal()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to be tested.

Author

L. M. Thompson

Date

2017

5.1.2.139 mqc_scalar_isallocated()

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

MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.140 mqc_scalar_sin()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar) The argument of the function.`

Author

L. M. Thompson

Date

2019

5.1.2.141 mqc_scalar_sqrt()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar) The argument of the function.`

Author

L. M. Thompson

Date

2016

5.1.2.142 mqc_scalar_tan()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.143 mqc_scalaradd()

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

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

Purpose:

```
MQC_ScalarAdd is a function that sums two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be summed.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be summed.

Author

L. M. Thompson

Date

2016

5.1.2.144 mqc_scalarcomplexadd()

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

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

Purpose:

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

Parameters

in	*ComplexIn*	Complex is Complex(kind=real64) The intrinsic complex variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum.

Author

L. M. Thompson

Date

2019

5.1.2.145 mqc_scalarcomplexdivide()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	*ComplexIn*	ComplexIn is Complex(kind=real64) The intrinsic complex variable denominator.

Author

L. M. Thompson

Date

2019

5.1.2.146 mqc_scalarcomplexexponent()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar) The base value.`
in	*Comp↩ In*	`CompIn is Complex(kind=real64) The power value.`

Author

L. M. Thompson

Date

2019

5.1.2.147 mqc_scalarcomplexmultiply()

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

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

Purpose:

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

Parameters

in	*Complex↩ In*	`Complex is Complex(kind=real64) The intrinsic complex variable to multiply.`
in	*Scalar*	`Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.148 mqc_scalarcomplexsubtract()

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

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

Purpose:

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from.
in	*ComplexIn*	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract.

Author

L. M. Thompson

Date

2019

5.1.2.149 mqc_scalardivide()

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

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

Purpose:

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

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The numerator.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The denominator.

Author

L. M. Thompson

Date

2016

5.1.2.150 mqc_scalareq()

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

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

Purpose:

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

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.2.151 mqc_scalarexponent()

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

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

Purpose:

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

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The base value.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The power value.

Author

L. M. Thompson

Date

2016

5.1.2.152 mqc_scalarge()

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

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

Purpose:

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

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

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar that will be tested.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2016

5.1.2.153 mqc_scalargt()

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

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

Purpose:

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

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

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar that will be tested.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2016

5.1.2.154 mqc_scalargtinteger()

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

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

Purpose:

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

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`
in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer that will be tested.`

Author

L. M. Thompson

Date

2019

5.1.2.155 mqc_scalargtreal()

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

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

Purpose:

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

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

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	*Real↩ In*	RealIn is Real(kind=int64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

5.1.2.156 mqc_scalarintegeradd()

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

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

Purpose:

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

Parameters

in	*Integer↵ In*	IntegerIn is Integer(kind=int64) The intrinsic integer variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar varibale to sum.

Author

L. M. Thompson

Date

2019

5.1.2.157 mqc_scalarintegerdivide()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable numerator.`
in	*Integer↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable denominator.`

Author

L. M. Thompson

Date

2019

5.1.2.158 mqc_scalarintegerexponent()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar)` `The base value.`
in	*IntIn*	`IntIn is Integer(kind=int64)` `The power value.`

Author

L. M. Thompson

Date

2019

5.1.2.159 mqc_scalarintegermultiply()

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

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

Purpose:

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

Parameters

in	*IntegerIn*	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar varibale to multiply.

Author

L. M. Thompson

Date

2019

5.1.2.160 mqc_scalarintegersubtract()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract from.`
in	*Integer↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer to subtract.`

Author

L. M. Thompson

Date

2019

5.1.2.161 mqc_scalarle()

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

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

Purpose:

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

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

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar that will be tested.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2016

5.1.2.162 mqc_scalarleinteger()

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

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

Purpose:

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

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.163 mqc_scalarlereal()

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

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

Purpose:

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

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`
in	*Real↵ In*	`RealIn is Real(kind=int64)` `The intrinsic real that will be tested.`

Author

L. M. Thompson

Date

2019

5.1.2.164 mqc_scalarlt()

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

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

Purpose:

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

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

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar that will be tested.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2016

5.1.2.165 mqc_scalarltreal()

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

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

Purpose:

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

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real that will be tested.`

Author

L. M. Thompson

Date

2019

5.1.2.166 mqc_scalarmatrixproduct()

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

5.1.2.167 mqc_scalarmultiply()

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

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

Purpose:

```
MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar to be multiplied.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar to be multiplied.

Author

L. M. Thompson

Date

2016

5.1.2.168 mqc_scalarne()

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

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

Purpose:

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

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

5.1.2.169 mqc_scalarrealadd()

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

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

Purpose:

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

Parameters

in	*realIn*	`RealIn is Real(kind=real64)` `The intrinsic real variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to sum.`

Author

L. M. Thompson

Date

2019

5.1.2.170 mqc_scalarrealdivide()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable numerator.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real variable denominator.`

Author

L. M. Thompson

Date

2019

5.1.2.171 mqc_scalarrealexponent()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar)` `The base value.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The power value.`

Author

L. M. Thompson

Date

2019

5.1.2.172 mqc_scalarrealmultiply()

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

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

Purpose:

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

Parameters

in	*realIn*	`RealIn is Real(kind=real64)` `The intrinsic real variable to multiply.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.173 mqc_scalarrealsubtract()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract from.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real to subtract.`

Author

L. M. Thompson

Date

2019

5.1.2.174 mqc_scalarsubtract()

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

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

Purpose:

```
MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar from which Scalar2 will be subtracted.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar which will be subtracted from Scalar1.

Author

L. M. Thompson

Date

2016

5.1.2.175 mqc_scalarvectordifference()

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

MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector

Purpose:

```
MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from
all elements of an MQC vector.
```

Parameters

in	*Scalar↩ In*	`ScalarIn is Type(MQC_Scalar)`
		`The MQC scalar to be subtracted from elements of the`
		`the MQC vector.`

in	*Vector↩ In*	`VectorIn is Type(MQC_Vector)`
		`The MQC vector with elements from which ScalarIn will`
		`be subtracted.`

Author

L. M. Thompson

Date

2016

5.1.2.176 mqc_scalarvectorproduct()

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

MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector

Purpose:

```
MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar
with a MQC vector.
```

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar to multiply.`
in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`

Author

X. Sheng

A. D. Mahler

Date

2017, 2019

5.1.2.177 mqc_scalarvectorsum()

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

MQC_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector

Purpose:

```
MQC_VectorVectorSum is a function that adds an MQC scalar to all elements of
an MQC vector.
```

Parameters

in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The MQC scalar to add to the MQC vector.
in	*VectorIn*	VectorIn is Type(MQC_Vector) The MQC vector with elements to sum with ScalarIn.

Author

L. M. Thompson

Date

2016

5.1.2.178 mqc_set_array2tensor()

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

5.1.2.179 mqc_set_array2vector_complex()

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

MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic
complex array equal to a MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.180 mqc_set_array2vector_integer()

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

MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic
integer array equal to a MQC vector
```

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.181 mqc_set_array2vector_real()

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

MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic
real array equal to a MQC vector.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.182 mqc_set_complexarray2matrix()

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

MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array

Purpose:

```
MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic complex rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.183 mqc_set_integerarray2matrix()

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

MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array

Purpose:

```
MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic integer rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.184 mqc_set_matrix2complexarray()

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

MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex
rank-2 array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.185 mqc_set_matrix2integerarray()

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

MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer
rank-2 array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.186 mqc_set_matrix2matrix()

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

MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix

Purpose:

```
MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another
MQC matrix.
```

Parameters

in,out	*MatrixOut*	`MatrixOut is Class(MQC_Matrix)` `The MQC matrix to be set equal to the incoming MQC matrix.`
in	*MatrixIn*	`MatrixIn is Class(MQC_Matrix)` `The MQC matrix to be input into MatrixOut.`

Author

L. M. Thompson

Date

2016

5.1.2.187 mqc_set_matrix2realarray()

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

MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2
array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.188 mqc_set_realarray2matrix()

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

MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array

Purpose:

```
MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic real rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.189 mqc_set_vector2complexarray()

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

MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array

Purpose:

```
MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic complex array.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.190 mqc_set_vector2integerarray()

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

MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array

Purpose:

```
MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic integer array.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.191 mqc_set_vector2realarray()

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

MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array

Purpose:

```
MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic real array.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.192 mqc_set_vector2vector()

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

MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector

Purpose:

```
MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another
MQC vector.
```

Parameters

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

Author

H. P. Hratchian

L. M. Thompson

Date

2016

5.1.2.193 mqc_vector2diagmatrix()

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

MQC_Vector2DiagMatrix is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector

Purpose:

```
MQC_Vector2DiagMatrix is a function that outputs a diagonal MQC matrix with
elements defined by an MQC vector.
```

Parameters

in	*Vector*	`Vector is Class(MQC_Vector)` `MQC vector defining diagonal elements of output matrix.`

Author

X. Sheng

Date

2017

5.1.2.194 mqc_vector_abs()

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

MQC_Vector_Abs is a function that returns the absolute value of all elements of an MQC vector

Purpose:

```
MQC_Vector_Sqrt is a function that returns the absolute value of all elements
of an MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.195 mqc_vector_argsort()

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

MQC_Vector_Argsort is a function that returns the indices of an an MQC vector sorted from low to high

Purpose:

```
MQC_Vector_Argsort is a function that returns the indices of an MQC vector
sorted from low to high.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.196 mqc_vector_cast_complex()

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

MQC_vector_cast_complex is a function that converts an MQC vector to its complex space

Purpose:

```
MQC_vector_cast_complex is a function that converts an MQC vector to its
complex space.
```

Parameters

in	*VA*	VA is Class(MQC_Vector) The MQC vector to convert.

Author

L. M. Thompson

Date

2017

5.1.2.197 mqc_vector_cast_integer()

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

MQC_vector_cast_integer is a function that converts an MQC vector to its integer space

Purpose:

```
MQC_vector_cast_integer is a function that converts an MQC vector to its
integer space.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.198 mqc_vector_cast_real()

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

MQC_vector_cast_real is a function that converts an MQC vector to its real space

Purpose:

```
MQC_vector_cast_real is a function that converts an MQC vector to its real
space.
```

Parameters

in	*VA*	VA is Class(MQC_Vector) The MQC vector to convert.

Author

X. Sheng

Date

2017

5.1.2.199 mqc_vector_cmplx()

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

MQC_Vector_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

Purpose:

```
MQC_Vector_Cmplx is a function that takes a MQC vector representing the real
part and a MQC vector representing the imaginary part and combines them into
another MQC vector.
```

Parameters

in	*Vector1*	Vector1 is Type(MQC_Vector) The MQC vector containing the real part.
in	*Vector2*	Vector2 is Type(MQC_Vector) The MQC vector containing the imaginary part.

Author

L. M. Thompson

Date

2019

5.1.2.200 mqc_vector_complex_imagpart()

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

MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

Purpose:

```
MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with
elements containing the imaginary part of elements of another MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.201 mqc_vector_complex_realpart()

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

MQC_Vector_Complex_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

Purpose:

```
MQC_Vector_Complex_RealPart is a function that returns a MQC vector with
elements containing the real part of elements of another MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.202 mqc_vector_conjugate_transpose()

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

MQC_Vector_Conjugate_Transpose is a function that returns the conjugate transpose of an MQC vector

Purpose:

```
MQC_Vector_Conjugate_Transpose is a function that returns the conjugate
transpose of an MQC vector.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC vector to conjugate transpose.

Author

L. M. Thompson

Date

2017

5.1.2.203 mqc_vector_copy_complex2int()

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

MQC_Vector_Copy_Complex2Int is a subroutine that copies a complex MQC_Vector into its integer vector

Purpose:

```
MQC_Vector_Copy_Complex2Int is a subroutine that copies a complex MQC_Vector
into its integer vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.204 mqc_vector_copy_complex2real()

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

MQC_Vector_Copy_Complex2Real is a subroutine that copies a complex MQC_Vector into its real vector

Purpose:

```
MQC_Vector_Copy_Complex2Real is a subroutine that copies a complex MQC_Vector
into its real vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.205 mqc_vector_copy_int2complex()

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

MQC_Vector_Copy_Int2Complex is a subroutine that copies an integer MQC_Vector into its complex vector

Purpose:

```
MQC_Vector_Copy_Int2Complex is a subroutine that copies an integer MQC_Vector
into its complex vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.206 mqc_vector_copy_int2real()

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

MQC_Vector_Copy_Int2Real is a subroutine that copies an integer MQC_Vector into its real vector

Purpose:

```
MQC_Vector_Copy_Int2Real is a subroutine that copies an integer MQC_Vector
into its real vector.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.207 mqc_vector_copy_real2complex()

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

MQC_Vector_Copy_Real2Complex is a subroutine that copies a real MQC_Vector into its complex vector

Purpose:

```
MQC_Vector_Copy_Real2Complex is a subroutine that copies a real MQC_Vector
into its complex vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.208 mqc_vector_copy_real2int()

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

MQC_Vector_Copy_Real2Int is a subroutine that copies a real MQC_Vector into its integer vector

Purpose:

```
MQC_Vector_Copy_Real2Int is a subroutine that copies a real MQC_Vector into its
integer vector.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.209 mqc_vector_havecomplex()

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

MQC_Vector_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.210 mqc_vector_haveinteger()

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

MQC_Vector_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector

Purpose:

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

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.211 mqc_vector_havereal()

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

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

Purpose:

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

Parameters

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

Author

H. P. Hratchian

Date

2016

5.1.2.212 mqc_vector_initialize()

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

MQC_Vector_Initialize is a subroutine that initializes a MQC vector

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.213 mqc_vector_isallocated()

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

MQC_Vector_isAllocated is a function that returns TRUE is an MQC vector is allocated and FALSE if it is not

Purpose:

```
MQC_Vector_isAllocated is a function that returns TRUE is an MQC vector is
allocated and FALSE if it is not.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.214 mqc_vector_iscolumn()

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

MQC_Vector_IsColumn is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector

Purpose:

```
MQC_Vector_IsColumn is a function that returns TRUE if the MQC vector is a
column vector and FALSE if the MQC vector is a row vector.
```

Parameters

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

Author

L. M. Thompson

Date

2016

5.1.2.215 mqc_vector_maxloc()

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

MQC_Vector_MaxLoc is a function that returns the index of the largest value in an MQC vector

Purpose:

```
MQC_Vector_MaxLoc is a function that returns the index of the largest value in
an MQC vector.
```

Parameters

in	*Vector*	`Vector is Class(MQC_Vector)` `The name of the MQC_Vector variable.`

Author

L. M. Thompson

Date

2019

5.1.2.216 mqc_vector_maxval()

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

MQC_Vector_MaxVal is a function that returns the largest value in an MQC vector

Purpose:

```
MQC_Vector_MaxVal is a function that returns the largest value in an MQC
vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.217 mqc_vector_minloc()

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

MQC_Vector_MinLoc is a function that returns the index of the smallest value in an MQC vector

Purpose:

```
MQC_Vector_MinLoc is a function that returns the index of the smallest value
in an MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.218 mqc_vector_minval()

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

MQC_Vector_MinVal is a function that returns the smallest value in an MQC vector

Purpose:

```
MQC_Vector_MinVal is a function that returns the smallest value in an MQC
vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.219 mqc_vector_norm()

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

MQC_Vector_Norm is a function that returns the norm of an MQC vector

Purpose:

```
MQC_Vector_Norm is a function that returns the norm of an MQC vector. The
following options are available:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.220 mqc_vector_pop()

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

MQC_Vector_Pop is a function that removes a value from the end of a MQC vector and returns it

Purpose:

```
MQC_Vector_Pop is a function that removes a value from the end of a MQC vector
and returns it.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.221 mqc_vector_power()

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

MQC_Vector_Power is a function that returns the value of all elements of an MQC vector raised to a power

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.222 mqc_vector_push()

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

MQC_Vector_Push is a function that adds a value to the end of a MQC vector

Purpose:

```
MQC_Vector_Push is a function that adds a value to the end of a MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.223 mqc_vector_scalar_at()

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

MQC_Vector_Scalar_At is a function that returns the ith element of a MQC vector as an MQC scalar

Purpose:

```
MQC_Vector_Scalar_At is a function that returns the ith element of a MQC vector
as an MQC scalar. If the location of the element is negative the it is counted
from the end of MQC vector.
```

Parameters

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

Author

X. Sheng

Date

2017

5.1.2.224 mqc_vector_scalar_increment()

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

MQC_Vector_Scalar_Increment is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar

Purpose:

```
MQC_Vector_Scalar_Increment is a subroutine that increments the value of the
ith element of a MQC vector by the value of a MQC scalar. If the location of
the element is negative then it is counted from the end of MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.225 mqc_vector_scalar_put()

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

MQC_Vector_Scalar_Put is a subroutine that updates the value of the ith element of a MQC vector with the value of a MQC scalar

Purpose:

```
MQC_Vector_Scalar_Put is a subroutine that updates the value of the ith element
of a MQC vector with the value of a MQC scalar. If the location of the element
is negative then it is counted from the end of MQC vector.
```

Parameters

in	*Vector*	`Vector is Class(MQC_Vector)` `The MQC_Vector to update at the Ith element.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar to update Vector at element I.`

in	*I*	`I is Integer(kind=int64)`
		`The location of the element in Vector to update. If I`
		`is negative it is counted from the last element of`
		`Vector.`

Author

L. M. Thompson

Date

2017

5.1.2.226 mqc_vector_shift()

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

MQC_Vector_Shift is a function that removes a value from the beginning of a MQC vector and returns it

Purpose:

```
MQC_Vector_Pop is a function that removes a value from the beginning of a MQC
vector and returns it.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.227 mqc_vector_sort()

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

MQC_Vector_Sort is a function that returns an MQC vector sorted from low to high unless optional index order is present

Purpose:

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

Parameters

in,out	*Vector*	`Vector is Class(MQC_Vector)` `The name of the MQC_Vector variable.`
in	*idx*	`idx is Type(MQC_Vector),Optional` `The new order of indices after sort.`

Author

L. M. Thompson

Date

2019

5.1.2.228 mqc_vector_sqrt()

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

MQC_Vector_Sqrt is a function that returns the square root of all elements of an MQC vector

Purpose:

```
MQC_Vector_Sqrt is a function that returns the square root of all elements of
an MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.229 mqc_vector_transpose()

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

MQC_Vector_Transpose is a function that returns the transpose of an MQC vector

Purpose:

```
MQC_Vector_Transpose is a function that returns the transpose of an MQC vector.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC vector to transpose.

Author

H. P. Hratchian

Date

2016

5.1.2.230 mqc_vector_unshift()

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

MQC_Vector_Unshift is a function that adds a value to the beginning of a MQC vector

Purpose:

```
MQC_Vector_Unshift is a function that adds a value to the beginning of a MQC
vector.
```

Parameters

in,out	*Vector*	Vector is Class(MQC_Vector) The name of the MQC_Vector variable.
in	*Scalar*	Scalar is Type(MQC_Scalar) The value that will be added to the beginning of Vector.

Author

L. M. Thompson

Date

2017

5.1.2.231 mqc_vector_vector_at()

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

MQC_Vector_Vector_At is a function that returns the vector at the specified subvector of MQC_Vector

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.232 mqc_vector_vector_put()

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

MQC_Vector_Vector_Put is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector

Purpose:

```
MQC_Vector_Vector_Put is a subroutine that updates the values of a subvector
of a MQC vector starting at element I with the values of a MQC vector. Negative
I counts from the end of vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

5.1.2.233 mqc_vectorcomplexdivide()

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

MQC_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar

Purpose:

```
MQC_VectorComplexDivide is a function that returns a MQC vector divided by an
intrinsic complex scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to divide.`
in	*Comp← In*	`CompIn is Complex(kind=comp64)` `The intrinsic complex scalar to divide by.`

Author

L. M. Thompson

Date

2019

5.1.2.234 mqc_vectorcomplexproduct()

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

MQC_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar

Purpose:

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

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`
in	*Comp← In*	`CompIn is Complex(kind=real64)` `The integer complex to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.235 mqc_vectorintegerdivide()

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

MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar

Purpose:

```
MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an
intrinsic integer scalar.
```

Parameters

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

Author

L. M. Thompson

Date

2019

5.1.2.236 mqc_vectorintegerproduct()

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

MQC_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar

Purpose:

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

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	*IntIn*	IntIn is Integer(kind=int64) The integer intrinsic to multiply.

Author

L. M. Thompson

Date

2019

5.1.2.237 mqc_vectormatrixdotproduct()

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

5.1.2.238 mqc_vectorrealdivide()

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

MQC_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer

Purpose:

```
MQC_VectorRealDivide is a function that returns a MQC vector divided by an
intrinsic real scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to divide.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real scalar to divide by.`

Author

L. M. Thompson

Date

2019

5.1.2.239 mqc_vectorrealproduct()

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

MQC_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar

Purpose:

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

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The real intrinsic to multiply.`

Author

L. M. Thompson

Date

2019

5.1.2.240 mqc_vectorscalardivide()

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

MQC_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar

Purpose:

```
MQC_VectorScalarDivide is a function that returns a MQC vector divided by a
MQC scalar.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to divide by.

Author

A. D. Mahler

Date

2019

5.1.2.241 mqc_vectorscalarproduct()

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

MQC_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar

Purpose:

```
MQC_VectorScalarProduct is a function that returns the product of a MQC vector
with a MQC scalar.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.

Author

X. Sheng

Date

2017

5.1.2.242 mqc_vectorvectordifference()

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

MQC_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector

Purpose:

```
MQC_VectorVectorDifference is a function that subtracts two MQC vectors and
stores them in another MQC vector.
```

Parameters

in	*Vector1↩ In*	Vector1In is Type(MQC_Vector) The first MQC vector from which the second will be subtracted.
in	*Vector2↩ In*	Vector2In is Type(MQC_Vector) The second MQC vector that will be subtracted from the first.

Author

L. M. Thompson

Date

2016

5.1.2.243 mqc_vectorvectordotproduct()

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

MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors

Purpose:

```
MQC_VectorVectorDotProduct is a function that returns the dot product of two
MQC vectors. The first vector should be a row vector, while the second vector
should be a column vector. The vectors should be of the same length.
```

Parameters

in	*Vector1*	Vector1 is Type(MQC_Vector) The MQC row vector.
in	*Vector2*	Vector2 is Type(MQC_Vector) The MQC column vector.

Author

H. P. Hratchian

L. M. Thompson

Date

2016

5.1.2.244 mqc_vectorvectorsum()

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

MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector

Purpose:

```
MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in
another MQC vector.
```

Parameters

in	*Vector1↵ In*	Vector1In is Type(MQC_Vector) The first MQC vector that will be summed.
in	*Vector2↵ In*	Vector2In is Type(MQC_Vector) The second MQC vector that will be summed.

Author

L. M. Thompson

Date

2016

5.1.2.245 symindexhash()

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

SymIndexHash is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor

Purpose:

```
SymIndexHash is a function that returns the index in a vector of a symmetric-
packed matrix or rank-4 tensor. If a matrix is tested, it is assumed lower-
triangular row-wise stored.
```

Parameters

in	*I*	I is integer(kind=int64) The first index of the matrix/rank-4 tensor.
in	*J*	I is integer(kind=int64) The second index of the matrix/rank-4 tensor.
in	*K*	K is integer(kind=int64),Optional The third index of the rank-4 tensor. This argument is only required when taking hash of a rank-4 tensor.
in	*L*	L is integer(kind=int64),Optional The fourth index of the rank-4 tensor. This argument is only required when taking hash of a rank-4 tensor.

Author

X. Sheng

L. M. Thompson

Date

2017

5.2 mqc_est Module Reference

Data Types

- interface assignment(=)

- interface contraction
- interface dagger
- interface dot_product
- interface matmul
- type mqc_determinant
- type mqc_determinant_string
- interface mqc_matrix_undospinblockghf
- interface mqc_print
- type mqc_pscf_wavefunction
- type mqc_scf_eigenvalues
- type mqc_scf_integral
- type mqc_twoeris
- type mqc_wavefunction
- interface operator(∗)
- interface operator(+)
- interface operator(-)
- interface transpose

Functions/Subroutines

- subroutine mqc_print_wavefunction (wavefunction, iOut, label)
- subroutine mqc_print_integral (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_print_eigenvalues (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_print_twoeris (twoERIs, iOut, header, blank_at_top, blank_at_bottom)
- logical function mqc_integral_isallocated (Integral)
- logical function mqc_eigenvalues_isallocated (Eigenvalues)
- logical function mqc_integral_has_alpha (integral)
- logical function mqc_integral_has_beta (integral)
- logical function mqc_integral_has_alphabeta (integral)
- logical function mqc_integral_has_betaalpha (integral)
- logical function mqc_eigenvalues_has_alpha (eigenvalues)
- logical function mqc_eigenvalues_has_beta (eigenvalues)
- character(len=64) function mqc_integral_array_type (integral)
- character(len=64) function mqc_eigenvalues_array_type (eigenvalues)
- character(len=64) function mqc_integral_array_name (integral)
- character(len=64) function mqc_eigenvalues_array_name (eigenvalues)
- subroutine mqc_integral_add_name (integral, arrayName)
- subroutine mqc_eigenvalues_add_name (eigenvalues, arrayName)
- integer(kind=int64) function mqc_integral_dimension (integral, label, axis)
- integer(kind=int64) function mqc_eigenvalues_dimension (eigenvalues, label)
- subroutine mqc_twoeris_allocate (twoERIs, storageType, integralType, alpha, beta, alphaBeta, betaAlpha)
- subroutine mqc_integral_allocate (integral, arrayName, arrayType, alpha, beta, alphaBeta, betaAlpha)
- subroutine mqc_eigenvalues_allocate (eigenvalues, arrayName, arrayType, alpha, beta)
- subroutine mqc_integral_identity (integral, nAlpha, nBeta, label, nAlpha2, nBeta2)
- subroutine mqc_integral_initialize (integral, nAlpha, nBeta, scalar, label, nAlpha2, nBeta2)
- type(mqc_matrix) function mqc_integral_output_block (integral, blockName)
- type(mqc_scf_integral) function mqc_integral_output_orbitals (integral, orbString, alphaOrbsIn, betaOrbsIn, axis)
- type(mqc_scf_integral) function mqc_integral_swap_orbitals (integral, alphaOrbsIn, betaOrbsIn, axis)
- type(mqc_vector) function mqc_eigenvalues_output_block (eigenvalues, blockName)
- subroutine mqc_integral_output_array (matrixOut, integralIn)

- 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 (integralIn, 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, lEVecs)
- subroutine mqc_scf_integral_diagonalize (integral, eVals, eVecs)
- type(mqc_scf_integral) function mqc_scf_integral_inverse (integral)
- type(mqc_scalar) function mqc_scf_integral_trace (integral)
- type(mqc_scalar) function mqc_scf_integral_determinant (integral)
- subroutine mqc_integral_set_energy_list (integral, elist)
- integer(kind=int64) function, dimension(:), allocatable mqc_integral_get_energy_list (integral)
- subroutine mqc_integral_delete_energy_list (integral)
- subroutine mqc_scf_eigenvalues_power (eigenvalues, power)
- type(mqc_scalar) function mqc_twoeris_at (twoERIs, i, j, k, l, spinBlock)
- type(mqc_scalar) function mqc_integral_at (integral, i, j, spinBlock)
- type(mqc_scalar) function mqc_eigenvalues_at (eigenvalues, i, spinBlock)
- subroutine mqc_scf_transformation_matrix (overlap, transform_matrix, nBasUse)
- subroutine gen_det_str (IOut, IPrint, NBasisIn, NAlphaIn, NBetaIn, Determinants, NCoreIn)
- type(mqc_scalar) function slater_condon (IOut, IPrint, NBasisIn, Determinants, L_A_String, L_B_String, R_A_↩ String, R_B_String, Core_Hamiltonian, ERIs, UHF)
- subroutine twoeri_trans (IOut, IPrint, MO_Coeff, ERIs, MO_ERIs, UHF)
- subroutine mqc_build_ci_hamiltonian (IOut, IPrint, NBasis, Determinants, MO_Core_Ham, MO_ERIs, UHF, CI↩ _Hamiltonian)
- type(mqc_matrix) function get_one_gamma_matrix (iOut, iPrint, nBasisIn, nState, determinants, ci_amplitudes, nCoreIn, nOrbsIn)

5.2.1 Function/Subroutine Documentation

5.2.1.1 gen_det_str()

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

5.2.1.2 get_one_gamma_matrix()

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

5.2.1.3 mqc_build_ci_hamiltonian()

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

5.2.1.4 mqc_eigenvalue_eigenvalue_dotproduct()

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

5.2.1.5 mqc_eigenvalues_add_name()

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

5.2.1.6 mqc_eigenvalues_allocate()

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

5.2.1.7 mqc_eigenvalues_array_name()

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

5.2.1.8 mqc_eigenvalues_array_type()

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

5.2.1.9 mqc_eigenvalues_at()

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

5.2.1.10 mqc_eigenvalues_dimension()

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

5.2.1.11 mqc_eigenvalues_eigenvalues_multiply()

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

5.2.1.12 mqc_eigenvalues_has_alpha()

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

5.2.1.13 mqc_eigenvalues_has_beta()

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

5.2.1.14 mqc_eigenvalues_integral_multiply()

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

5.2.1.15 mqc_eigenvalues_isallocated()

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

5.2.1.16 mqc_eigenvalues_output_array()

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

5.2.1.17 mqc_eigenvalues_output_block()

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

5.2.1.18 mqc_eri_integral_contraction()

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

5.2.1.19 mqc_integral_add_name()

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

5.2.1.20 mqc_integral_allocate()

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

5.2.1.21 mqc_integral_array_name()

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

5.2.1.22 mqc_integral_array_type()

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

5.2.1.23 mqc_integral_at()

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

5.2.1.24 mqc_integral_conjugate_transpose()

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

5.2.1.25 mqc_integral_delete_energy_list()

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

5.2.1.26 mqc_integral_difference()

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

5.2.1.27 mqc_integral_dimension()

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

5.2.1.28 mqc_integral_eigenvalues_multiply()

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

5.2.1.29 mqc_integral_get_energy_list()

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

5.2.1.30 mqc_integral_has_alpha()

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

5.2.1.31 mqc_integral_has_alphabeta()

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

5.2.1.32 mqc_integral_has_beta()

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

5.2.1.33 mqc_integral_has_betaalpha()

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

5.2.1.34 mqc_integral_identity()

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

5.2.1.35 mqc_integral_initialize()

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

5.2.1.36 mqc_integral_integral_multiply()

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

5.2.1.37 mqc_integral_isallocated()

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

5.2.1.38 mqc_integral_matrix_multiply()

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

5.2.1.39 mqc_integral_norm()

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

5.2.1.40 mqc_integral_output_array()

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

5.2.1.41 mqc_integral_output_block()

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

5.2.1.42 mqc_integral_output_orbitals()

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

5.2.1.43 mqc_integral_scalar_multiply()

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

5.2.1.44 mqc_integral_set_energy_list()

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

5.2.1.45 mqc_integral_sum()

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

5.2.1.46 mqc_integral_swap_orbitals()

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

5.2.1.47 mqc_integral_transpose()

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

5.2.1.48 mqc_matrix_integral_multiply()

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

5.2.1.49 mqc_matrix_spinblockghf()

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

5.2.1.50 mqc_matrix_undospinblockghf_eigenvalues()

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

5.2.1.51 mqc_matrix_undospinblockghf_integral()

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

5.2.1.52 mqc_print_eigenvalues()

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

5.2.1.53 mqc_print_integral()

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

5.2.1.54 mqc_print_twoeris()

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

5.2.1.55 mqc_print_wavefunction()

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

5.2.1.56 mqc_scalar_integral_multiply()

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

5.2.1.57 mqc_scf_eigenvalues_power()

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

5.2.1.58 mqc_scf_integral_contraction()

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

5.2.1.59 mqc_scf_integral_determinant()

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

5.2.1.60 mqc_scf_integral_diagonalize()

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

5.2.1.61 mqc_scf_integral_generalized_eigensystem()

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

5.2.1.62 mqc_scf_integral_inverse()

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

5.2.1.63 mqc_scf_integral_trace()

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

5.2.1.64 mqc_scf_transformation_matrix()

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

5.2.1.65 mqc_twoeris_allocate()

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

5.2.1.66 mqc_twoeris_at()

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

5.2.1.67 slater_condon()

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

5.2.1.68 twoeri_trans()

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

Chapter 6

Data Type Documentation

6.1 mqc_algebra::abs Interface Reference

Takes the absolute value

Public Member Functions

- type(mqc_scalar) function mqc_scalar_get_abs_value (Scalar)
  MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable
- type(mqc_vector) function mqc_vector_abs (A)
  MQC_Vector_Abs is a function that returns the absolute value of all elements of an MQC vector

6.1.1 Detailed Description

Takes the absolute value

6.1.2 Member Function/Subroutine Documentation

6.1.2.1 mqc_scalar_get_abs_value()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Class(MQC_Scalar)` `The MQC_Scalar to be tested.`

Author

A. Mahler

Date

2018

6.1.2.2 mqc_vector_abs()

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

MQC_Vector_Abs is a function that returns the absolute value of all elements of an MQC vector

Purpose:

```
MQC_Vector_Sqrt is a function that returns the absolute value of all elements
of an MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.2 mqc_algebra::acos Interface Reference

Returns the arccosine

Public Member Functions

- type(mqc_scalar) function mqc_scalar_acos (Scalar)

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

6.2.1 Detailed Description

Returns the arccosine

6.2.2 Member Function/Subroutine Documentation

6.2.2.1 mqc_scalar_acos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.3 mqc_algebra::aimag Interface Reference

Returns the imaginary part

Public Member Functions

- type(mqc_scalar) function mqc_scalar_complex_imagpart (ScalarIn)

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

  MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

6.3.1 Detailed Description

Returns the imaginary part

6.3.2 Member Function/Subroutine Documentation

6.3.2.1 mqc_scalar_complex_imagpart()

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

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

Purpose:

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

Parameters

in	*Scalar↩ In*	`ScalarIn is Type(MQC_Scalar)` `The MQC_Scalar input variable.`

Author

L. M. Thompson

Date

2019

6.3.2.2 mqc_vector_complex_imagpart()

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

MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector

Purpose:

```
MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with
elements containing the imaginary part of elements of another MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.4 mqc_algebra::asin Interface Reference

Returns the arcsine

Public Member Functions

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

6.4.1 Detailed Description

Returns the arcsine

6.4.2 Member Function/Subroutine Documentation

6.4.2.1 mqc_scalar_asin()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.5 mqc_algebra::assignment(=) Interface Reference

Assigns a variable to the value of another

Public Member Functions

- subroutine mqc_input_integer_scalar (ScalarOut, ScalarIn)
  MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an MQC_Scalar
- subroutine mqc_input_real_scalar (ScalarOut, ScalarIn)
  MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar
- subroutine mqc_input_complex_scalar (ScalarOut, ScalarIn)
  MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an MQC_Scalar
- subroutine mqc_output_mqcscalar_scalar (ScalarOut, ScalarIn)
  MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar
- subroutine mqc_output_integer_scalar (ScalarOut, ScalarIn)
  MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar
- subroutine mqc_output_real_scalar (ScalarOut, ScalarIn)
  MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar
- subroutine mqc_output_complex_scalar (ScalarOut, ScalarIn)
  MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar
- subroutine mqc_set_vector2vector (VectorOut, VectorIn)
  MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector
- subroutine mqc_set_vector2integerarray (ArrayOut, VectorIn)
  MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array
- subroutine mqc_set_vector2realarray (ArrayOut, VectorIn)
  MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array
- subroutine mqc_set_vector2complexarray (ArrayOut, VectorIn)
  MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array
- subroutine mqc_set_array2vector_integer (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector
- subroutine mqc_set_array2vector_real (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector
- subroutine mqc_set_array2vector_complex (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector
- subroutine mqc_set_matrix2matrix (MatrixOut, MatrixIn)
  MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix

- subroutine mqc_set_matrix2integerarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix
- subroutine mqc_set_matrix2realarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix
- subroutine mqc_set_matrix2complexarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix
- subroutine mqc_set_integerarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array
- subroutine mqc_set_realarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array
- subroutine mqc_set_complexarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array
- subroutine mqc_set_array2tensor (TensorOut, ArrayIn)

6.5.1 Detailed Description

Assigns a variable to the value of another

6.5.2 Member Function/Subroutine Documentation

6.5.2.1 mqc_input_complex_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.2 mqc_input_integer_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Type(MQC_Scalar) The name of the output variable.
in	*ScalarIn*	ScalarIn is Integer(kind=int64) The value of the input variable.

Author

L. M. Thompson

Date

2016

6.5.2.3 mqc_input_real_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Type(MQC_Scalar) The name of the output variable.
in	*ScalarIn*	ScalarIn is Real(kind=real64) The value of the input variable.

Author

L. M. Thompson

Date

2016

6.5.2.4 mqc_output_complex_scalar()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.5 mqc_output_integer_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Integer(kind=int64) The name of the output variable.
in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The value of the input variable.

Author

L. M. Thompson

Date

2016

6.5.2.6 mqc_output_mqcscalar_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Type(MQC_Scalar) The name of the output variable.
in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The value of the input variable.

Author

L. M. Thompson

Date

2016

6.5.2.7 mqc_output_real_scalar()

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

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

Purpose:

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

Parameters

in,out	*ScalarOut*	ScalarOut is Real(kind=real64) The name of the output variable.
in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The value of the input variable.

Author

L. M. Thompson

Date

2016

6.5.2.8 mqc_set_array2tensor()

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

6.5.2.9 mqc_set_array2vector_complex()

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

MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic
complex array equal to a MQC vector.
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.10 mqc_set_array2vector_integer()

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

MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic
integer array equal to a MQC vector
```

Parameters

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

Author

H. P. Hratchian

Date

2016

6.5.2.11 mqc_set_array2vector_real()

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

MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic
real array equal to a MQC vector.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

6.5.2.12 mqc_set_complexarray2matrix()

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

MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array

Purpose:

```
MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic complex rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.13 mqc_set_integerarray2matrix()

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

MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array

Purpose:

```
MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic integer rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.2.14 mqc_set_matrix2complexarray()

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

MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex
rank-2 array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.15 mqc_set_matrix2integerarray()

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

MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer
rank-2 array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.2.16 mqc_set_matrix2matrix()

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

MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix

Purpose:

```
MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another
MQC matrix.
```

Parameters

in,out	*MatrixOut*	`MatrixOut is Class(MQC_Matrix)` `The MQC matrix to be set equal to the incoming MQC matrix.`
in	*MatrixIn*	`MatrixIn is Class(MQC_Matrix)` `The MQC matrix to be input into MatrixOut.`

Author

L. M. Thompson

Date

2016

6.5.2.17 mqc_set_matrix2realarray()

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

MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix

Purpose:

```
MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2
array equal to an MQC matrix.
```

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.2.18 mqc_set_realarray2matrix()

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

MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array

Purpose:

```
MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an
intrinsic real rank-2 array.
```

Parameters

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

Author

L. M. Thompson

Date

2016

6.5.2.19 mqc_set_vector2complexarray()

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

MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array

Purpose:

```
MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic complex array.
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.5.2.20 mqc_set_vector2integerarray()

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

MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array

Purpose:

```
MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic integer array.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

6.5.2.21 mqc_set_vector2realarray()

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

MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array

Purpose:

```
MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to
a rank 1 intrinsic real array.
```

Parameters

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

Author

H. P. Hratchian

Date

2016

6.5.2.22 mqc_set_vector2vector()

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

MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector

Purpose:

```
MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another
MQC vector.
```

Parameters

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

Author

H. P. Hratchian

L. M. Thompson

Date

2016

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

- src/mqc_algebra.F03

6.6 mqc_est::assignment(=) Interface Reference

Public Member Functions

- subroutine mqc_integral_output_array (matrixOut, integralIn)
- subroutine mqc_eigenvalues_output_array (vectorOut, eigenvaluesIn)

6.6.1 Member Function/Subroutine Documentation

6.6.1.1 mqc_eigenvalues_output_array()

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

6.6.1.2 mqc_integral_output_array()

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

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

- src/mqc_est.F03

6.7 mqc_algebra::atan Interface Reference

Returns the arctangent

Public Member Functions

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

6.7.1 Detailed Description

Returns the arctangent

6.7.2 Member Function/Subroutine Documentation

6.7.2.1 mqc_scalar_atan()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The argument of the function.`

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.8 mqc_algebra::atan2 Interface Reference

Returns the arctangent accounting for circle quadrant

Public Member Functions

- type(mqc_scalar) function mqc_scalar_atan2 (Scalar)

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

6.8.1 Detailed Description

Returns the arctangent accounting for circle quadrant

6.8.2 Member Function/Subroutine Documentation

6.8.2.1 mqc_scalar_atan2()

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

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

Purpose:

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

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The argument of the function.`

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.9 mqc_algebra::cmplx Interface Reference

Defines a complex number

Public Member Functions

- type(mqc_scalar) function mqc_scalar_cmplx (Scalar1, Scalar2)

  MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars
- type(mqc_vector) function mqc_vector_cmplx (Vector1, Vector2)

  MQC_Vector_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

6.9.1 Detailed Description

Defines a complex number

6.9.2 Member Function/Subroutine Documentation

6.9.2.1 mqc_scalar_cmplx()

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

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

Purpose:

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

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The real part of MQC_Scalar_Cmplx.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The imaginary part of MQC_Scalar_Cmplx.`

Author

L. M. Thompson

Date

2019

6.9.2.2 mqc_vector_cmplx()

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

MQC_Vector_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

Purpose:

```
MQC_Vector_Cmplx is a function that takes a MQC vector representing the real
part and a MQC vector representing the imaginary part and combines them into
another MQC vector.
```

Parameters

in	*Vector1*	`Vector1 is Type(MQC_Vector)` `The MQC vector containing the real part.`
in	*Vector2*	`Vector2 is Type(MQC_Vector)` `The MQC vector containing the imaginary part.`

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.10 mqc_algebra::conjg Interface Reference

Returns the complex conjugate

Public Member Functions

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

6.10.1 Detailed Description

Returns the complex conjugate

6.10.2 Member Function/Subroutine Documentation

6.10.2.1 mqc_scalar_complex_conjugate()

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

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

Purpose:

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

Parameters

in	*ScalarIn*	`ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.`

Author

L. M. Thompson

Date

2018

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

- src/mqc_algebra.F03

6.11 mqc_algebra::contraction Interface Reference

Contracts two arrays

Public Member Functions

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

6.11.1 Detailed Description

Contracts two arrays

6.11.2 Member Function/Subroutine Documentation

6.11.2.1 mqc_matrix_matrix_contraction()

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

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

- src/mqc_algebra.F03

6.12 mqc_est::contraction Interface Reference

Public Member Functions

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

6.12.1 Member Function/Subroutine Documentation

6.12.1.1 mqc_eri_integral_contraction()

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

6.12.1.2 mqc_scf_integral_contraction()

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

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

- src/mqc_est.F03

6.13 mqc_algebra::cos Interface Reference

Returns the cosine

Public Member Functions

- type(mqc_scalar) function mqc_scalar_cos (Scalar)

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

6.13.1 Detailed Description

Returns the cosine

6.13.2 Member Function/Subroutine Documentation

6.13.2.1 mqc_scalar_cos()

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

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

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.14 mqc_algebra::dagger Interface Reference

Returns the Hermitian conjugate

Public Member Functions

- type(mqc_vector) function mqc_vector_conjugate_transpose (Vector)

  MQC_Vector_Conjugate_Transpose is a function that returns the conjugate transpose of an MQC vector
- type(mqc_matrix) function mqc_matrix_conjugate_transpose (Matrix)

  MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate transpose of a MQC matrix

6.14.1 Detailed Description

Returns the Hermitian conjugate

6.14.2 Member Function/Subroutine Documentation

6.14.2.1 mqc_matrix_conjugate_transpose()

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

MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate transpose of a MQC matrix

Purpose:

```
MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate
transpose of a MQC matrix.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be conjugate transposed.

Author

L. M. Thompson

Date

2016

6.14.2.2 mqc_vector_conjugate_transpose()

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

MQC_Vector_Conjugate_Transpose is a function that returns the conjugate transpose of an MQC vector

Purpose:

```
MQC_Vector_Conjugate_Transpose is a function that returns the conjugate
transpose of an MQC vector.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC vector to conjugate transpose.`

Author

L. M. Thompson

Date

2017

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

- src/mqc_algebra.F03

6.15 mqc_est::dagger Interface Reference

Public Member Functions

- type(mqc_scf_integral) function mqc_integral_conjugate_transpose (integral, label)

6.15.1 Member Function/Subroutine Documentation

6.15.1.1 mqc_integral_conjugate_transpose()

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

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

- src/mqc_est.F03

6.16 mqc_algebra::dot_product Interface Reference

Returns the dot product

Public Member Functions

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

    MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors

6.16.1 Detailed Description

Returns the dot product

6.16.2 Member Function/Subroutine Documentation

6.16.2.1 mqc_vectorvectordotproduct()

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

MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors

Purpose:

```
MQC_VectorVectorDotProduct is a function that returns the dot product of two
MQC vectors. The first vector should be a row vector, while the second vector
should be a column vector. The vectors should be of the same length.
```

Parameters

in	*Vector1*	Vector1 is Type(MQC_Vector) The MQC row vector.
in	*Vector2*	Vector2 is Type(MQC_Vector) The MQC column vector.

Author

H. P. Hratchian

L. M. Thompson

Date

2016

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

- src/mqc_algebra.F03

6.17 mqc_est::dot_product Interface Reference

Public Member Functions

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

6.17.1 Member Function/Subroutine Documentation

6.17.1.1 mqc_eigenvalue_eigenvalue_dotproduct()

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

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

- src/mqc_est.F03

6.18 mqc_algebra::matmul Interface Reference

Multiplies two arrays

Public Member Functions

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

6.18.1 Detailed Description

Multiplies two arrays

6.18.2 Member Function/Subroutine Documentation

6.18.2.1 mqc_matrixmatrixdotproduct()

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

6.18.2.2 mqc_matrixvectordotproduct()

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

6.18.2.3 mqc_vectormatrixdotproduct()

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

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

- src/mqc_algebra.F03

6.19 mqc_est::matmul Interface Reference

Public Member Functions

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

6.19.1 Member Function/Subroutine Documentation

6.19.1.1 mqc_eigenvalues_eigenvalues_multiply()

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

6.19.1.2 mqc_eigenvalues_integral_multiply()

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

6.19.1.3 mqc_integral_eigenvalues_multiply()

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

6.19.1.4 mqc_integral_integral_multiply()

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

6.19.1.5 mqc_integral_matrix_multiply()

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

6.19.1.6 mqc_matrix_integral_multiply()

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

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

- src/mqc_est.F03

6.20 mqc_algebra::matrix_symm2sq Interface Reference

Sets a symmetric packed intrinsic array as a square packed intrinsic array

Public Member Functions

- subroutine matrix_symm2sq_integer (N, I_Symm, I_Sq)

  Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array
- subroutine matrix_symm2sq_real (N, A_Symm, A_Sq)

  Matrix_Symm2Sq_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array
- subroutine matrix_symm2sq_complex (N, A_Symm, A_Sq)

  Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array

6.20.1 Detailed Description

Sets a symmetric packed intrinsic array as a square packed intrinsic array

6.20.2 Member Function/Subroutine Documentation

6.20.2.1 matrix_symm2sq_complex()

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

Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array

Purpose:

```
Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric-packed
intrinsic complex matrix to a rank-2 complex array.
TODO: Move this routine to MQC general
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.20.2.2 matrix_symm2sq_integer()

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

Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array

Purpose:

```
Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric-packed
intrinsic integer matrix to a rank-2 integer array.
TODO: Move this routine to MQC general
```

Parameters

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

Author

H. P. Hratchian

Date

2017

6.20.2.3 matrix_symm2sq_real()

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

Matrix_Symm2Sq_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array

Purpose:

```
Matrix_Symm2Sq_Real is a subroutine that converts a symmetric-packed
intrinsic real matrix to a rank-2 real array.
TODO: Move this routine to MQC general
```

Parameters

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

Author

H. P. Hratchian

Date

2017

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

- src/mqc_algebra.F03

6.21 mqc_algebra::mqc_cast_complex Interface Reference

Sets an array to complex type

Public Member Functions

- type(mqc_vector) function mqc_vector_cast_complex (VA)
  MQC_vector_cast_complex is a function that converts an MQC vector to its complex space
- type(mqc_matrix) function mqc_matrix_cast_complex (MA)
  MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its complex space

6.21.1 Detailed Description

Sets an array to complex type

6.21.2 Member Function/Subroutine Documentation

6.21.2.1 mqc_matrix_cast_complex()

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

MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its complex space

Purpose:

```
MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its
complex space.
```

Parameters

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

Author

L. M. Thompson

Date

2017

6.21.2.2 mqc_vector_cast_complex()

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

MQC_vector_cast_complex is a function that converts an MQC vector to its complex space

Purpose:

```
MQC_vector_cast_complex is a function that converts an MQC vector to its
complex space.
```

Parameters

in	*VA*	`VA is Class(MQC_Vector)` `The MQC vector to convert.`

Author

L. M. Thompson

Date

2017

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

- src/mqc_algebra.F03

6.22 mqc_algebra::mqc_cast_integer Interface Reference

Sets an array to integer type

Public Member Functions

- type(mqc_vector) function mqc_vector_cast_integer (VA)

    MQC_vector_cast_integer is a function that converts an MQC vector to its integer space
- type(mqc_matrix) function mqc_matrix_cast_integer (MA)

    MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its integer space

6.22.1 Detailed Description

Sets an array to integer type

6.22.2 Member Function/Subroutine Documentation

6.22.2.1 mqc_matrix_cast_integer()

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

MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its integer space

Purpose:

```
MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its
integer space.
```

Parameters

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

Author

L. M. Thompson

Date

2019

6.22.2.2 mqc_vector_cast_integer()

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

MQC_vector_cast_integer is a function that converts an MQC vector to its integer space

Purpose:

```
MQC_vector_cast_integer is a function that converts an MQC vector to its
integer space.
```

Parameters

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

Author

L. M. Thompson

Date

2019

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

- src/mqc_algebra.F03

6.23 mqc_algebra::mqc_cast_real Interface Reference

Sets an array to real type

Public Member Functions

- type(mqc_vector) function mqc_vector_cast_real (VA)
  MQC_vector_cast_real is a function that converts an MQC vector to its real space
- type(mqc_matrix) function mqc_matrix_cast_real (MA)
  MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real space

6.23.1 Detailed Description

Sets an array to real type

6.23.2 Member Function/Subroutine Documentation

6.23.2.1 mqc_matrix_cast_real()

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

MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real space

Purpose:

```
MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real
space.
```

Parameters

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

Author

X. Sheng

Date

2017

6.23.2.2 mqc_vector_cast_real()

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

MQC_vector_cast_real is a function that converts an MQC vector to its real space

Purpose:

```
MQC_vector_cast_real is a function that converts an MQC vector to its real
space.
```

Parameters

in	*VA*	`VA is Class(MQC_Vector)` `The MQC vector to convert.`

Author

X. Sheng

Date

2017

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

- src/mqc_algebra.F03

6.24 mqc_est::mqc_determinant Type Reference

Public Attributes

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

6.24.1 Member Data Documentation

6.24.1.1 nalpstr

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

6.24.1.2 nbetstr

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

6.24.1.3 ndets

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

6.24.1.4 order

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

6.24.1.5 strings

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

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

- src/mqc_est.F03

6.25 mqc_est::mqc_determinant_string Type Reference

Public Attributes

- type(mqc_matrix) alpha
- type(mqc_matrix) beta

6.25.1 Member Data Documentation

6.25.1.1 alpha

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

6.25.1.2 beta

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

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

- src/mqc_est.F03

6.26 mqc_algebra::mqc_have_complex Interface Reference

Determines in an array is complex type

Public Member Functions

- logical function mqc_vector_havecomplex (Vector)

  MQC_Vector_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector
- logical function mqc_matrix_havecomplex (Matrix)

  MQC_Matrix_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix

6.26.1 Detailed Description

Determines in an array is complex type

6.26.2 Member Function/Subroutine Documentation

6.26.2.1 mqc_matrix_havecomplex()

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

MQC_Matrix_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix

Purpose:

```
MQC_Matrix_HaveComplex is a function that returns TRUE if an MQC matrix has an
allocated complex matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2017

6.26.2.2 mqc_vector_havecomplex()

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

MQC_Vector_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector

Purpose:

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

Parameters

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

Author

L. M. Thompson

Date

2017

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

- src/mqc_algebra.F03

6.27 mqc_algebra::mqc_have_int Interface Reference

Determines in an array is integer type

Public Member Functions

- logical function mqc_vector_haveinteger (Vector)

  MQC_Vector_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector
- logical function mqc_matrix_haveinteger (Matrix)

  MQC_Matrix_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix

6.27.1 Detailed Description

Determines in an array is integer type

6.27.2 Member Function/Subroutine Documentation

6.27.2.1 mqc_matrix_haveinteger()

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

MQC_Matrix_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix

Purpose:

```
MQC_Matrix_HaveInteger is a function that returns TRUE if an MQC matrix has an
allocated integer matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2016

6.27.2.2 mqc_vector_haveinteger()

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

MQC_Vector_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector

Purpose:

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

Parameters

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

Author

H. P. Hratchian

Date

2016

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

- src/mqc_algebra.F03

6.28 mqc_algebra::mqc_have_real Interface Reference

Determines in an array is real type

Public Member Functions

- logical function mqc_vector_havereal (Vector)

  MQC_Vector_HaveReal is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector
- logical function mqc_matrix_havereal (Matrix)

  MQC_Matrix_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix

6.28.1 Detailed Description

Determines in an array is real type

6.28.2 Member Function/Subroutine Documentation

6.28.2.1 mqc_matrix_havereal()

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

MQC_Matrix_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix

Purpose:

```
MQC_Matrix_HaveReal is a function that returns TRUE if an MQC matrix has an
allocated real matrix and FALSE if it does not.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be tested.

Author

L. M. Thompson

Date

2016

6.28.2.2 mqc_vector_havereal()

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

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

Purpose:

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

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to be tested.`

Author

H. P. Hratchian

Date

2016

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

- src/mqc_algebra.F03

6.29 mqc_algebra::mqc_matrix Type Reference

Rank 2 array variable

Public Member Functions

- Procedure, public print => mqc_print_matrix_algebra1
  Print the MQC Matrix
- Procedure, public init => mqc_matrix_initialize
  Initialize the MQC Matrix with a specified value
- Procedure, public identity => mqc_matrix_identity
  Initialize the MQC Matrix as the identity matrix
- Procedure, public set => mqc_matrix_set
  Set all elements in the MQC Matrix to a specified value
- Procedure, public norm => mqc_matrix_norm
  Returns the norm of the MQC Matrix
- Procedure, public transpose => mqc_matrix_transpose
  Returns the transpose of the MQC Matrix
- Procedure, public dagger => mqc_matrix_conjugate_transpose
  Returns the Hemitian transpose of the MQC Matrix
- Procedure, public diag => mqc_matrix_diagonalize
  Returns the eigenvalues and eigenvalues of a symmetric or hermitian MQC Matrix
- Procedure, public svd => mqc_matrix_svd
  Computes the singular value decomposition of the MQC Matrix
- Procedure, public eigensys => mqc_matrix_generalized_eigensystem

Solves the generalized eigenproblem of the MQC Matrix
- Procedure, public inv => mqc_matrix_inverse

  Returns the inverse of the MQC Matrix
- Procedure, public det => mqc_matrix_determinant

  Returns the determinant of the MQC Matrix
- Procedure, public trace => mqc_matrix_trace

  Returns the trace of the MQC Matrix
- Procedure, public rmsmax => mqc_matrix_rms_max

  Returns the root mean square deviation and maximum deviation of elements of the MQC Matrix
- Procedure, public sqrt => mqc_matrix_sqrt

  Returns the square root of the MQC Matrix elements
- Procedure, public at => mqc_matrix_scalar_at

  Returns the value of the specified element of the MQC Matrix
- Procedure, public vat => mqc_matrix_vector_at

  Returns the vector of the specified subvector of the MQC Matrix
- Procedure, public mat => mqc_matrix_matrix_at

  Returns the matrix of the specified submatrix of the MQC Matrix
- Procedure, public put => mqc_matrix_scalar_put

  Updates the specified element of the MQC Matrix to the specified value
- Procedure, public vput => mqc_matrix_vector_put

  Updates the specified subvector of the MQC Matrix to the specified vector
- Procedure, public mput => mqc_matrix_matrix_put

  Updates the specified submatrix of the MQC Matrix to the specified matrix

6.29.1 Detailed Description

Rank 2 array variable

6.29.2 Member Function/Subroutine Documentation

6.29.2.1 at()

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

Returns the value of the specified element of the MQC Matrix

6.29.2.2 dagger()

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

Returns the Hemitian transpose of the MQC Matrix

6.29.2.3 det()

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

Returns the determinant of the MQC Matrix

6.29.2.4 diag()

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

Returns the eigenvalues and eigenvalues of a symmetric or hermitian MQC Matrix

6.29.2.5 eigensys()

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

Solves the generalized eigenproblem of the MQC Matrix

6.29.2.6 identity()

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

Initialize the MQC Matrix as the identity matrix

6.29.2.7 init()

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

Initialize the MQC Matrix with a specified value

6.29.2.8 inv()

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

Returns the inverse of the MQC Matrix

6.29.2.9 mat()

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

Returns the matrix of the specified submatrix of the MQC Matrix

6.29.2.10 mput()

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

Updates the specified submatrix of the MQC Matrix to the specified matrix

6.29.2.11 norm()

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

Returns the norm of the MQC Matrix

6.29.2.12 print()

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

Print the MQC Matrix

6.29.2.13 put()

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

Updates the specified element of the MQC Matrix to the specified value

6.29.2.14 rmsmax()

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

Returns the root mean square deviation and maximum deviation of elements of the MQC Matrix

6.29.2.15 set()

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

Set all elements in the MQC Matrix to a specified value

6.29.2.16 sqrt()

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

Returns the square root of the MQC Matrix elements

6.29.2.17 svd()

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

Computes the singular value decomposition of the MQC Matrix

6.29.2.18 trace()

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

Returns the trace of the MQC Matrix

6.29.2.19 transpose()

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

Returns the transpose of the MQC Matrix

6.29.2.20 vat()

`Procedure, public mqc_algebra::mqc_matrix::vat ( )`

Returns the vector of the specified subvector of the MQC Matrix

6.29.2.21 vput()

```
Procedure, public mqc_algebra::mqc_matrix::vput ( )
```

Updates the specified subvector of the MQC Matrix to the specified vector

The documentation for this type was generated from the following file:

- src/mqc_algebra.F03

6.30 mqc_algebra::mqc_matrix_diagmatrix_put Interface Reference

Sets a diagonal packed intinsic array as an MQC Matrix object

Public Member Functions

- subroutine mqc_matrix_diagmatrix_put_integer (mat, diagMatrixIn)

  MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_matrix_diagmatrix_put_real (mat, diagMatrixIn)

  MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_matrix_diagmatrix_put_complex (mat, diagMatrixIn)

  MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector
- subroutine mqc_matrix_diagmatrix_put_vector (diagVectorIn, mat)

  MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

6.30.1 Detailed Description

Sets a diagonal packed intinsic array as an MQC Matrix object

6.30.2 Member Function/Subroutine Documentation

6.30.2.1 mqc_matrix_diagmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_complex (
            class(mqc_matrix), intent(inout) mat,
            complex(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic complex vector.
```

Parameters

in,out	*mat*	`Mat is class(MQC_Matrix)` `MQC matrix to overwrite with output diagonal matrix.`
in	*Diag←MatrixIn*	`DiagMatrixIn is complex(kind=real64),dimension(:)` `Intrinsic complex vector to write as diagonal matrix.`

Author

L. M. Thompson

Date

2017

6.30.2.2 mqc_matrix_diagmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_integer (
            class(mqc_matrix), intent(inout) mat,
            integer(kind=int64), dimension(:), intent(in) diagMatrixIn )
```

MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic integer vector.
```

Parameters

in,out	*mat*	`Mat is class(MQC_Matrix)` `MQC matrix to overwrite with output diagonal matrix.`
in	*Diag←MatrixIn*	`DiagMatrixIn is integer(kind=int64),dimension(:)` `Intrinsic integer vector to write as diagonal matrix.`

Author

H. P. Hratchian

Date

2017

6.30.2.3 mqc_matrix_diagmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_real (
            class(mqc_matrix), intent(inout) mat,
            real(kind=real64), dimension(:), intent(in) diagMatrixIn )
```

MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC
matrix with elements defined by values in an intrinsic real vector.
```

Parameters

in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.
in	*Diag↵ MatrixIn*	DiagMatrixIn is real(kind=real64),dimension(:) Intrinsic real vector to write as diagonal matrix.

Author

H. P. Hratchian

Date

2017

6.30.2.4 mqc_matrix_diagmatrix_put_vector()

```
subroutine mqc_algebra::mqc_matrix_diagmatrix_put::mqc_matrix_diagmatrix_put_vector (
            class(mqc_vector), intent(in) diagVectorIn,
            class(mqc_matrix), intent(inout) mat )
```

MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector

Purpose:

```
MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC
matrix with elements defined by values in a MQC vector.
```

Parameters

in	*Diag↔ VectorIn*	DiagVectorIn is class(MQC_Vector) Name of the MQC vector to write as diagonal matrix.
in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output diagonal matrix.

Author

L. M. Thompson

Date

2018

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.31 mqc_algebra::mqc_matrix_symmmatrix_put Interface Reference

Sets a symmetric packed intrinsic array as an MQC Matrix object

Public Member Functions

- subroutine mqc_matrix_symmmatrix_put_integer (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_matrix_symmmatrix_put_real (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_matrix_symmmatrix_put_complex (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector

6.31.1 Detailed Description

Sets a symmetric packed intrinsic array as an MQC Matrix object

6.31.2 Member Function/Subroutine Documentation

6.31.2.1 mqc_matrix_symmmatrix_put_complex()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_complex (
            class(mqc_matrix), intent(inout) mat,
            complex(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic complex
vector.
```

Parameters

in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	*Symm↩ MatrixIn*	SymmMatrixIn is complex(kind=real64),dimension(:) Intrinsic complex vector to write as symmetric-packed matrix.

Author

L. M. Thompson

Date

2017

6.31.2.2 mqc_matrix_symmmatrix_put_integer()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_integer (
            class(mqc_matrix), intent(inout) mat,
            integer(kind=int64), dimension(:), intent(in) symmMatrixIn )
```

MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic integer
vector.
```

Parameters

in,out	*mat*	Mat is class(MQC_Matrix) MQC matrix to overwrite with output symmetric matrix.
in	*Symm↔ MatrixIn*	SymmMatrixIn is integer(kind=int64),dimension(:) Intrinsic integer vector to write as symmetric-packed matrix.

Author

H. P. Hratchian

Date

2017

6.31.2.3 mqc_matrix_symmmatrix_put_real()

```
subroutine mqc_algebra::mqc_matrix_symmmatrix_put::mqc_matrix_symmmatrix_put_real (
            class(mqc_matrix), intent(inout) mat,
            real(kind=real64), dimension(:), intent(in) symmMatrixIn )
```

MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector

Purpose:

```
MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric
packed MQC matrix with elements defined by values in an intrinsic real
vector.
```

Parameters

in,out	*mat*	`Mat is class(MQC_Matrix)` `MQC matrix to overwrite with output symmetric matrix.`
in	*Symm↵ MatrixIn*	`SymmMatrixIn is real(kind=real64),dimension(:)` `Intrinsic real vector to write as symmetric-packed matrix.`

Author

H. P. Hratchian

Date

2017

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.32 mqc_est::mqc_matrix_undospinblockghf Interface Reference

Public Member Functions

- subroutine mqc_matrix_undospinblockghf_eigenvalues (eigenvaluesIn, vectorOut)
- subroutine mqc_matrix_undospinblockghf_integral (integralIn, matrixOut)

6.32.1 Member Function/Subroutine Documentation

6.32.1.1 mqc_matrix_undospinblockghf_eigenvalues()

```
subroutine mqc_est::mqc_matrix_undospinblockghf::mqc_matrix_undospinblockghf_eigenvalues (
            type(mqc_scf_eigenvalues), intent(in) eigenvaluesIn,
            type(mqc_vector), intent(out) vectorOut )
```

6.32.1.2 mqc_matrix_undospinblockghf_integral()

```
subroutine mqc_est::mqc_matrix_undospinblockghf::mqc_matrix_undospinblockghf_integral (
            type(mqc_scf_integral), intent(in) integralIn,
            type(mqc_matrix), intent(out) matrixOut )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.33 mqc_algebra::mqc_print Interface Reference

Prints an object

Public Member Functions

- subroutine mqc_print_scalar_algebra1 (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)
  MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar
- subroutine mqc_print_vector_algebra1 (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At_Bottom)
  MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector
- subroutine mqc_print_matrix_algebra1 (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)
  MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix
- subroutine mqc_print_r4tensor_algebra1 (Tensor, IOut, Header, blank_at_top, blank_at_bottom)

6.33.1 Detailed Description

Prints an object

6.33.2 Member Function/Subroutine Documentation

6.33.2.1 mqc_print_matrix_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_matrix_algebra1 (
            class(mqc_matrix), intent(in) Matrix,
            integer(kind=int64), intent(in) IOut,
            character(len=*), intent(in) Header,
            logical, intent(in), optional Blank_At_Top,
            logical, intent(in), optional Blank_At_Bottom )
```

MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix

Purpose:

```
MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix.
Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank
lines before or after output.
```

Parameters

in	*Matrix*	`Matrix is Class(MQC_Matrix)` `The variable to be printed.`
in	*IOut*	`IOut is Integer(kind=int64)` `The Fortran file number to print to.`
in	*Header*	`Header is Character(Len=*)` `The title to print along with Matrix.`

in	*Blank_At_Top*	`Blank_At_Top is Logical,Optional`
		`= .True.:  print blank line above output`
		`= .False.: do not print blank line above output.`

in	*Blank_At_Bottom*	`Blank_At_Bottom is Logical,Optional`
		`= .True.:  print blank line below output`
		`= .False.: do not print blank line below output.`

Author

L. M. Thompson

Date

2016

6.33.2.2 mqc_print_r4tensor_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_r4tensor_algebra1 (
            class(mqc_r4tensor), intent(in) Tensor,
            integer(kind=int64), intent(in) IOut,
            character(len=*), intent(in), optional Header,
            logical, optional blank_at_top,
            logical, optional blank_at_bottom )
```

6.33.2.3 mqc_print_scalar_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_scalar_algebra1 (
            class(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IOut,
            character(len=*), intent(in) Header,
            logical, intent(in), optional Blank_At_Top,
            logical, intent(in), optional Blank_At_Bottom )
```

MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar

Purpose:

```
MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar. Blank_At_Top
and Blank_At_Bottom are optional logical arguments to print blank lines before or
after output.
```

Parameters

in	*Scalar*	`Scalar is Class(MQC_Scalar)` `The variable to be printed.`
in	*IOut*	`IOut is Integer(kind=int64)` `The Fortran file number to print to.`
in	*Header*	`Header is Character(Len=*)` `The title to print along with Scalar.`

in	*Blank_At_Top*	`Blank_At_Top is Logical,Optional`
		`= .True.:  print blank line above output`
		`= .False.: do not print blank line above output.`

in	*Blank_At_Bottom*	`Blank_At_Bottom is Logical,Optional`
		`= .True.:  print blank line below output`
		`= .False.: do not print blank line below output.`

Author

L. M. Thompson

Date

2016

6.33.2.4 mqc_print_vector_algebra1()

```
subroutine mqc_algebra::mqc_print::mqc_print_vector_algebra1 (
            class(mqc_vector), intent(in) Vector,
            integer(kind=int64), intent(in) IOut,
            character(len=*), intent(in) Header,
            logical, intent(in), optional Verbose,
            logical, intent(in), optional Blank_At_Top,
            logical, intent(in), optional Blank_At_Bottom )
```

MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector

Purpose:

```
MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector.
Blank_At_Top and Blank_At_Bottom are optional logical arguments to print blank
lines before or after output.
```

Parameters

in	*Vector*	Vector is Class(MQC_Vector) The variable to be printed.
in	*IOut*	IOut is Integer(kind=int64) The Fortran file number to print to.
in	*Header*	Header is Character(Len=*) The title to print along with Vector.
in	*Verbose*	Verbose is Logical,Optional Adds extra printing to output.
in	*Blank_At_Top*	Blank_At_Top is Logical,Optional = .True.: print blank line above output = .False.: do not print blank line above output.
in	*Blank_At_Bottom*	Blank_At_Bottom is Logical,Optional = .True.: print blank line below output = .False.: do not print blank line below output.

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.34 mqc_est::mqc_print Interface Reference

Public Member Functions

- subroutine mqc_print_wavefunction (wavefunction, iOut, label)
- subroutine mqc_print_integral (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_print_eigenvalues (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_print_twoeris (twoERIs, iOut, header, blank_at_top, blank_at_bottom)

6.34.1 Member Function/Subroutine Documentation

6.34.1.1 mqc_print_eigenvalues()

```
subroutine mqc_est::mqc_print::mqc_print_eigenvalues (
            class(mqc_scf_eigenvalues) eigenvalues,
            integer(kind=int64), intent(in) iOut,
            character(len=*), intent(in) header,
            logical, intent(in), optional blank_at_top,
            logical, intent(in), optional blank_at_bottom )
```

6.34.1.2 mqc_print_integral()

```
subroutine mqc_est::mqc_print::mqc_print_integral (
            class(mqc_scf_integral) integral,
            integer(kind=int64), intent(in) iOut,
            character(len=*), intent(in) header,
            logical, intent(in), optional blank_at_top,
            logical, intent(in), optional blank_at_bottom )
```

6.34.1.3 mqc_print_twoeris()

```
subroutine mqc_est::mqc_print::mqc_print_twoeris (
            class(mqc_twoeris) twoERIs,
            integer(kind=int64), intent(in) iOut,
            character(len=*), intent(in) header,
            logical, intent(in), optional blank_at_top,
            logical, intent(in), optional blank_at_bottom )
```

6.34.1.4 mqc_print_wavefunction()

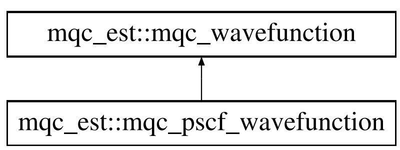
```
subroutine mqc_est::mqc_print::mqc_print_wavefunction (
            class(mqc_wavefunction) wavefunction,
            integer(kind=int64), intent(in) iOut,
            character(len=*), intent(in), optional label )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.35 mqc_est::mqc_pscf_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_pscf_wavefunction:



Public Attributes

- integer(kind=int64) ncore
- integer(kind=int64) nval
- integer(kind=int64) nactive
- integer(kind=int64) nfrz
- type(mqc_matrix) pscf_amplitudes
- type(mqc_vector) pscf_energies

Additional Inherited Members

6.35.1 Member Data Documentation

6.35.1.1 nactive

`integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nactive`

6.35.1.2 ncore

`integer(kind=int64) mqc_est::mqc_pscf_wavefunction::ncore`

6.35.1.3 nfrz

`integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nfrz`

6.35.1.4 nval

`integer(kind=int64) mqc_est::mqc_pscf_wavefunction::nval`

6.35.1.5 pscf_amplitudes

`type(mqc_matrix) mqc_est::mqc_pscf_wavefunction::pscf_amplitudes`

6.35.1.6 pscf_energies

`type(mqc_vector) mqc_est::mqc_pscf_wavefunction::pscf_energies`

The documentation for this type was generated from the following file:

- src/mqc_est.F03

6.36 mqc_algebra::mqc_r4tensor Type Reference

Updates the specified element of the MQC Matrix to the specified value

Public Member Functions

- Procedure, public print => mqc_print_r4tensor_algebra1

  Print the MQC R4Tensor
- Procedure, public at => mqc_r4tensor_at

  Return the specified element in the MQC R4Tensor
- Procedure, public put => mqc_r4tensor_put

  Update the specified element in the MQC R4Tensor with the specified value
- Procedure, public init => mqc_r4tensor_initialize

  Initialize the MQC R4Tensor

6.36.1 Detailed Description

Updates the specified element of the MQC Matrix to the specified value

Rank 4 array variable

6.36.2 Member Function/Subroutine Documentation

6.36.2.1 at()

`Procedure, public mqc_algebra::mqc_r4tensor::at ( )`

Return the specified element in the MQC R4Tensor

6.36.2.2 init()

`Procedure, public mqc_algebra::mqc_r4tensor::init ( )`

Initialize the MQC R4Tensor

6.36.2.3 print()

`Procedure, public mqc_algebra::mqc_r4tensor::print ( )`

Print the MQC R4Tensor

6.36.2.4 put()

`Procedure, public mqc_algebra::mqc_r4tensor::put ( )`

Update the specified element in the MQC R4Tensor with the specified value

The documentation for this type was generated from the following file:

- src/mqc_algebra.F03

6.37 mqc_algebra::mqc_scalar Type Reference

Rank 0 array variable

Public Member Functions

- Procedure, public print => mqc_print_scalar_algebra1
  Print the MQC Scalar
- Procedure, public rval => mqc_scalar_get_intrinsic_real
  Return the value of MQC Scalar as an intrinsic real
- Procedure, public ival => mqc_scalar_get_intrinsic_integer
  Return the value of MQC Scalar as an intrinsic integer
- Procedure, public cval => mqc_scalar_get_intrinsic_complex
  Return the value of MQC Scalar as an intrinsic complex
- Procedure, public abs => mqc_scalar_get_abs_value
  Take the absolute value of the MQC Scalar
- Procedure, public random => mqc_scalar_get_random_value
  Return a random value to the MQC Scalar

6.37.1 Detailed Description

Rank 0 array variable

6.37.2 Member Function/Subroutine Documentation

6.37.2.1 abs()

`Procedure, public mqc_algebra::mqc_scalar::abs ( )`

Take the absolute value of the MQC Scalar

6.37.2.2 cval()

```
Procedure, public mqc_algebra::mqc_scalar::cval ( )
```

Return the value of MQC Scalar as an intrinsic complex

6.37.2.3 ival()

```
Procedure, public mqc_algebra::mqc_scalar::ival ( )
```

Return the value of MQC Scalar as an intrinsic integer

6.37.2.4 print()

```
Procedure, public mqc_algebra::mqc_scalar::print ( )
```

Print the MQC Scalar

6.37.2.5 random()

```
Procedure, public mqc_algebra::mqc_scalar::random ( )
```

Return a random value to the MQC Scalar

6.37.2.6 rval()

```
Procedure, public mqc_algebra::mqc_scalar::rval ( )
```

Return the value of MQC Scalar as an intrinsic real

The documentation for this type was generated from the following file:

- src/mqc_algebra.F03

6.38 mqc_est::mqc_scf_eigenvalues Type Reference

Public Member Functions

- Procedure, public print => mqc_print_eigenvalues
- Procedure, public getlabel => mqc_eigenvalues_array_name
- Procedure, public addlabel => mqc_eigenvalues_add_name
- Procedure, public getblock => mqc_eigenvalues_output_block
- Procedure, public power => mqc_scf_eigenvalues_power
- Procedure, public at => mqc_eigenvalues_at

6.38.1 Member Function/Subroutine Documentation

6.38.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::addlabel ( )
```

6.38.1.2 at()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::at ( )
```

6.38.1.3 getblock()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::getblock ( )
```

6.38.1.4 getlabel()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::getlabel ( )
```

6.38.1.5 power()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::power ( )
```

6.38.1.6 print()

```
Procedure, public mqc_est::mqc_scf_eigenvalues::print ( )
```

The documentation for this type was generated from the following file:

- src/mqc_est.F03

6.39 mqc_est::mqc_scf_integral Type Reference

Public Member Functions

- Procedure, public print => mqc_print_integral
- Procedure, public getlabel => mqc_integral_array_name
- Procedure, public addlabel => mqc_integral_add_name
- Procedure, public getblock => mqc_integral_output_block
- Procedure, public identity => mqc_integral_identity
- Procedure, public init => mqc_integral_initialize
- Procedure, public diag => mqc_scf_integral_diagonalize
- Procedure, public eigensys => mqc_scf_integral_generalized_eigensystem
- Procedure, public inv => mqc_scf_integral_inverse
- Procedure, public trace => mqc_scf_integral_trace
- Procedure, public det => mqc_scf_integral_determinant
- Procedure, public norm => mqc_integral_norm
- Procedure, public setelist => mqc_integral_set_energy_list
- Procedure, public getelist => mqc_integral_get_energy_list
- Procedure, public deleteelist => mqc_integral_delete_energy_list
- Procedure, public orbitals => mqc_integral_output_orbitals
- Procedure, public swap => mqc_integral_swap_orbitals

6.39.1 Member Function/Subroutine Documentation

6.39.1.1 addlabel()

```
Procedure, public mqc_est::mqc_scf_integral::addlabel ( )
```

6.39.1.2 deleteelist()

```
Procedure, public mqc_est::mqc_scf_integral::deleteelist ( )
```

6.39.1.3 det()

Procedure, public mqc_est::mqc_scf_integral::det ( )

6.39.1.4 diag()

Procedure, public mqc_est::mqc_scf_integral::diag ( )

6.39.1.5 eigensys()

Procedure, public mqc_est::mqc_scf_integral::eigensys ( )

6.39.1.6 getblock()

Procedure, public mqc_est::mqc_scf_integral::getblock ( )

6.39.1.7 getelist()

Procedure, public mqc_est::mqc_scf_integral::getelist ( )

6.39.1.8 getlabel()

Procedure, public mqc_est::mqc_scf_integral::getlabel ( )

6.39.1.9 identity()

Procedure, public mqc_est::mqc_scf_integral::identity ( )

6.39.1.10 init()

Procedure, public mqc_est::mqc_scf_integral::init ( )

6.39.1.11 inv()

Procedure, public mqc_est::mqc_scf_integral::inv ( )

6.39.1.12 norm()

Procedure, public mqc_est::mqc_scf_integral::norm ( )

6.39.1.13 orbitals()

Procedure, public mqc_est::mqc_scf_integral::orbitals ( )

6.39.1.14 print()

Procedure, public mqc_est::mqc_scf_integral::print ( )

6.39.1.15 setelist()

Procedure, public mqc_est::mqc_scf_integral::setelist ( )

6.39.1.16 swap()

Procedure, public mqc_est::mqc_scf_integral::swap ( )

6.39.1.17 trace()

```
Procedure, public mqc_est::mqc_scf_integral::trace ( )
```

The documentation for this type was generated from the following file:

- src/mqc_est.F03

6.40 mqc_algebra::mqc_set_array2vector Interface Reference

Sets an intrinsic array as an MQC Algebra object

Public Member Functions

- subroutine mqc_set_array2vector_integer (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector
- subroutine mqc_set_array2vector_real (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector
- subroutine mqc_set_array2vector_complex (VectorOut, ArrayIn)
  MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

6.40.1 Detailed Description

Sets an intrinsic array as an MQC Algebra object

6.40.2 Member Function/Subroutine Documentation

6.40.2.1 mqc_set_array2vector_complex()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_complex (
            type(mqc_vector), intent(inout) VectorOut,
            complex(kind=real64), dimension(:), intent(in) ArrayIn )
```

MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic
complex array equal to a MQC vector.
```

Parameters

in,out	*VectorOut*	`VectorOut is Type(MQC_Vector)`
		`The MQC vector that will be set equal to the rank 1`
		`intrinsic array`

in	*ArrayIn*	`ArrayOut is Complex(kind=real64),Dimension(:)`
		`The rank 1 intrinsic array whose data will be input`
		`into the MQC vector.`

Author

L. M. Thompson

Date

2017

6.40.2.2 mqc_set_array2vector_integer()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_integer (
            type(mqc_vector), intent(inout) VectorOut,
            integer(kind=int64), dimension(:), intent(in) ArrayIn )
```

MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic
integer array equal to a MQC vector
```

Parameters

in,out	*VectorOut*	`VectorOut is Type(MQC_Vector)`
		`The MQC vector that will be set equal to the rank 1`
		`intrinsic array`

in	*ArrayIn*	`ArrayOut is Integer(kind=int64),Dimension(:)`
		`The rank 1 intrinsic array whose data will be input`
		`into the MQC vector.`

Author

H. P. Hratchian

Date

2016

6.40.2.3 mqc_set_array2vector_real()

```
subroutine mqc_algebra::mqc_set_array2vector::mqc_set_array2vector_real (
            type(mqc_vector), intent(inout) VectorOut,
            real(kind=real64), dimension(:), intent(in) ArrayIn )
```

MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector

Purpose:

```
MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic
real array equal to a MQC vector.
```

Parameters

in,out	*VectorOut*	VectorOut is Type(MQC_Vector) The MQC vector that will be set equal to the rank 1 intrinsic array
in	*ArrayIn*	ArrayOut is Real(kind=real64),Dimension(:) The rank 1 intrinsic array whose data will be input into the MQC vector.

Author

H. P. Hratchian

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.41 mqc_est::mqc_twoeris Type Reference

Public Member Functions

- procedure, public print => mqc_print_twoeris

6.41.1 Member Function/Subroutine Documentation

6.41.1.1 print()

```
procedure, public mqc_est::mqc_twoeris::print ( )
```

The documentation for this type was generated from the following file:

- src/mqc_est.F03

6.42 mqc_algebra::mqc_vector Type Reference

Rank 1 array variable

Public Member Functions

- Procedure, public print => mqc_print_vector_algebra1

  Print the MQC Vector
- Procedure, public size => mqc_length_vector

  Returns the length of the MQC Vector
- Procedure, public init => mqc_vector_initialize

  Initilizes the MQC Vector
- Procedure, public norm => mqc_vector_norm

  Returns the norm of the MQC Vector
- Procedure, public transpose => mqc_vector_transpose

  Returns the transpose of the MQC Vector
- Procedure, public dagger => mqc_vector_conjugate_transpose

  Returns the Hermitian conjugate of the MQC Vector
- Procedure, public at => mqc_vector_scalar_at

  Returns the value at the specified element of the MQC Vector
- Procedure, public vat => mqc_vector_vector_at

  Returns the subvector between specified element of the MQC Vector
- Procedure, public put => mqc_vector_scalar_put

  Updates the specified element of the MQC_Vector with the specified value

- Procedure, public vput => mqc_vector_vector_put

  Updates the specified subvector of the MQC_Vector with the specified vector
- Procedure, public push => mqc_vector_push

  Appends the specified value to the end of the MQC_Vector
- Procedure, public unshift => mqc_vector_unshift

  Prepends the specified value to the beginning of the MQC_Vector
- Procedure, public pop => mqc_vector_pop

  Removes the last element of the MQC_Vector and returns the value
- Procedure, public shift => mqc_vector_shift

  Removes the first element of the MQC_Vector and returns the value
- Procedure, public maxval => mqc_vector_maxval

  Returns the maximum value in the MQC_Vector
- Procedure, public minval => mqc_vector_minloc

  Returns the minimum value in the MQC_Vector
- Procedure, public maxloc => mqc_vector_maxval

  Returns the location of the maximum value in the MQC_Vector
- Procedure, public minloc => mqc_vector_minloc

  Returns the location of the minimum value in the MQC_Vector
- Procedure, public argsort => mqc_vector_argsort

  Returns the indices of the MQC_Vector sorted from low to high
- Procedure, public sort => mqc_vector_sort

  Returns the MQC_Vector sorted from low to high unless vector specifying index order is provided
- Procedure, public sqrt => mqc_vector_sqrt

  Returns the square root of each element in the MQC_Vector
- Procedure, public abs => mqc_vector_abs

  Returns the absolute value of each element in the MQC_Vector
- Procedure, public power => mqc_vector_power

  Returns each element in the MQC_Vector raised to a specified power
- Procedure, public diag => mqc_matrix_diagmatrix_put_vector, mqc_vector2diagmatrix

  Returns a diagonal MQC Matrix with values specified by the MQC_Vector

6.42.1 Detailed Description

Rank 1 array variable

6.42.2 Member Function/Subroutine Documentation

6.42.2.1 abs()

```
Procedure, public mqc_algebra::mqc_vector::abs ( )
```

Returns the absolute value of each element in the MQC_Vector

6.42.2.2 argsort()

```
Procedure, public mqc_algebra::mqc_vector::argsort ( )
```

Returns the indices of the MQC_Vector sorted from low to high

6.42.2.3 at()

```
Procedure, public mqc_algebra::mqc_vector::at ( )
```

Returns the value at the specified element of the MQC Vector

6.42.2.4 dagger()

```
Procedure, public mqc_algebra::mqc_vector::dagger ( )
```

Returns the Hermitian conjugate of the MQC Vector

6.42.2.5 diag()

```
Procedure, public mqc_algebra::mqc_vector::diag ( )
```

Returns a diagonal MQC Matrix with values specified by the MQC_Vector

6.42.2.6 init()

```
Procedure, public mqc_algebra::mqc_vector::init ( )
```

Initilizes the MQC Vector

6.42.2.7 maxloc()

```
Procedure, public mqc_algebra::mqc_vector::maxloc ( )
```

Returns the location of the maximum value in the MQC_Vector

6.42.2.8 maxval()

```
Procedure, public mqc_algebra::mqc_vector::maxval ( )
```

Returns the maximum value in the MQC_Vector

6.42.2.9 minloc()

```
Procedure, public mqc_algebra::mqc_vector::minloc ( )
```

Returns the location of the minimum value in the MQC_Vector

6.42.2.10 minval()

```
Procedure, public mqc_algebra::mqc_vector::minval ( )
```

Returns the minimum value in the MQC_Vector

6.42.2.11 norm()

```
Procedure, public mqc_algebra::mqc_vector::norm ( )
```

Returns the norm of the MQC Vector

6.42.2.12 pop()

```
Procedure, public mqc_algebra::mqc_vector::pop ( )
```

Removes the last element of the MQC_Vector and returns the value

6.42.2.13 power()

```
Procedure, public mqc_algebra::mqc_vector::power ( )
```

Returns each element in the MQC_Vector raised to a specified power

6.42.2.14 print()

```
Procedure, public mqc_algebra::mqc_vector::print ( )
```

Print the MQC Vector

6.42.2.15 push()

```
Procedure, public mqc_algebra::mqc_vector::push ( )
```

Appends the specified value to the end of the MQC_Vector

6.42.2.16 put()

```
Procedure, public mqc_algebra::mqc_vector::put ( )
```

Updates the specified element of the MQC_Vector with the specified value

6.42.2.17 shift()

```
Procedure, public mqc_algebra::mqc_vector::shift ( )
```

Removes the first element of the MQC_Vector and returns the value

6.42.2.18 size()

```
Procedure, public mqc_algebra::mqc_vector::size ( )
```

Returns the length of the MQC Vector

6.42.2.19 sort()

```
Procedure, public mqc_algebra::mqc_vector::sort ( )
```

Returns the MQC_Vector sorted from low to high unless vector specifying index order is provided

6.42.2.20 sqrt()

```
Procedure, public mqc_algebra::mqc_vector::sqrt ( )
```

Returns the square root of each element in the MQC_Vector

6.42.2.21 transpose()

```
Procedure, public mqc_algebra::mqc_vector::transpose ( )
```

Returns the transpose of the MQC Vector

6.42.2.22 unshift()

```
Procedure, public mqc_algebra::mqc_vector::unshift ( )
```

Prepends the specified value to the beginning of the MQC_Vector

6.42.2.23 vat()

```
Procedure, public mqc_algebra::mqc_vector::vat ( )
```

Returns the subvector between specified element of the MQC Vector

6.42.2.24 vput()

```
Procedure, public mqc_algebra::mqc_vector::vput ( )
```

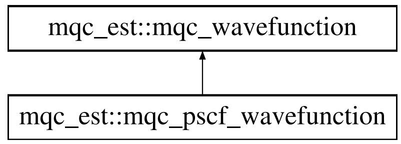
Updates the specified subvector of the MQC_Vector with the specified vector

The documentation for this type was generated from the following file:

- src/mqc_algebra.F03

6.43 mqc_est::mqc_wavefunction Type Reference

Inheritance diagram for mqc_est::mqc_wavefunction:



Public Member Functions

- Procedure, public print => mqc_print_wavefunction

Public Attributes

- type(mqc_scf_integral) mo_coefficients
- type(mqc_scf_eigenvalues) mo_energies
- type(mqc_scf_eigenvalues) mo_symmetries
- type(mqc_scf_integral) core_hamiltonian
- type(mqc_scf_integral) fock_matrix
- type(mqc_scf_integral) density_matrix
- type(mqc_scf_integral) scf_density_matrix
- type(mqc_scf_integral) overlap_matrix
- type(mqc_scalar) nalpha
- type(mqc_scalar) nbeta
- type(mqc_scalar) nelectrons
- type(mqc_scalar) nbasis
- type(mqc_scalar) charge
- type(mqc_scalar) multiplicity
- character(len=256) basis
- character(len=256) symmetry
- character(len=256) wf_type
- logical wf_complex

6.43.1 Member Function/Subroutine Documentation

6.43.1.1 print()

```
Procedure, public mqc_est::mqc_wavefunction::print ( )
```

6.43.2 Member Data Documentation

6.43.2.1 basis

`character(len=256) mqc_est::mqc_wavefunction::basis`

6.43.2.2 charge

`type(mqc_scalar) mqc_est::mqc_wavefunction::charge`

6.43.2.3 core_hamiltonian

`type(mqc_scf_integral) mqc_est::mqc_wavefunction::core_hamiltonian`

6.43.2.4 density_matrix

`type(mqc_scf_integral) mqc_est::mqc_wavefunction::density_matrix`

6.43.2.5 fock_matrix

`type(mqc_scf_integral) mqc_est::mqc_wavefunction::fock_matrix`

6.43.2.6 mo_coefficients

`type(mqc_scf_integral) mqc_est::mqc_wavefunction::mo_coefficients`

6.43.2.7 mo_energies

`type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_energies`

6.43.2.8 mo_symmetries

`type(mqc_scf_eigenvalues) mqc_est::mqc_wavefunction::mo_symmetries`

6.43.2.9 multiplicity

`type(mqc_scalar) mqc_est::mqc_wavefunction::multiplicity`

6.43.2.10 nalpha

`type(mqc_scalar) mqc_est::mqc_wavefunction::nalpha`

6.43.2.11 nbasis

`type(mqc_scalar) mqc_est::mqc_wavefunction::nbasis`

6.43.2.12 nbeta

`type(mqc_scalar) mqc_est::mqc_wavefunction::nbeta`

6.43.2.13 nelectrons

`type(mqc_scalar) mqc_est::mqc_wavefunction::nelectrons`

6.43.2.14 overlap_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::overlap_matrix
```

6.43.2.15 scf_density_matrix

```
type(mqc_scf_integral) mqc_est::mqc_wavefunction::scf_density_matrix
```

6.43.2.16 symmetry

```
character(len=256) mqc_est::mqc_wavefunction::symmetry
```

6.43.2.17 wf_complex

```
logical mqc_est::mqc_wavefunction::wf_complex
```

6.43.2.18 wf_type

```
character(len=256) mqc_est::mqc_wavefunction::wf_type
```

The documentation for this type was generated from the following file:

- src/mqc_est.F03

6.44 mqc_algebra::operator(*) Interface Reference

Multiplies two variables

Public Member Functions

- type(mqc_scalar) function mqc_scalarmultiply (Scalar1, Scalar2)

  MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects
- type(mqc_scalar) function mqc_integerscalarmultiply (IntegerIn, Scalar)

  MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegermultiply (Scalar, IntegerIn)

  MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_realscalarmultiply (RealIn, Scalar)

  MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealmultiply (Scalar, RealIn)

  MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_complexscalarmultiply (ComplexIn, Scalar)

  MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexmultiply (Scalar, ComplexIn)

  MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_vector) function mqc_scalarvectorproduct (Scalar, Vector)

  MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector
- type(mqc_vector) function mqc_vectorscalarproduct (vector, scalar)

  MQC_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar
- type(mqc_matrix) function mqc_scalarmatrixproduct (Scalar, Matrix)
- type(mqc_matrix) function mqc_matrixscalarproduct (Matrix, Scalar)
- type(mqc_vector) function mqc_realvectorproduct (RealIn, Vector)

  MQC_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector
- type(mqc_vector) function mqc_vectorrealproduct (vector, realIn)

  MQC_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar
- type(mqc_vector) function mqc_integervectorproduct (intIn, Vector)

  MQC_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector
- type(mqc_vector) function mqc_vectorintegerproduct (vector, intIn)

  MQC_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar
- type(mqc_vector) function mqc_complexvectorproduct (CompIn, Vector)

  MQC_ComplexVectorProduct is a function that returns the product of an intrinsic complex scalar and a MQC vector
- type(mqc_vector) function mqc_vectorcomplexproduct (vector, compIn)

  MQC_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar
- type(mqc_matrix) function mqc_matrixmatrixproduct (MA, MB)

  MQC_MatrixMatrixProduct is a function that computes the element- wise product of two MQC matrices

6.44.1 Detailed Description

Multiplies two variables

6.44.2 Member Function/Subroutine Documentation

6.44.2.1 mqc_complexscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_complexscalarmultiply (
            complex(kind=real64), intent(in) ComplexIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar

Purpose:

```
MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by
an MQC_Scalar.
```

Parameters

in	*ComplexIn*	Complex is Complex(kind=real64) The intrinsic complex variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.2 mqc_complexvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_complexvectorproduct (
            complex(kind=real64), intent(in) CompIn,
            type(mqc_vector), intent(in) Vector )
```

MQC_ComplexVectorProduct is a function that returns the product of an intrinsic complex scalar and a MQC vector

Purpose:

```
MQC_ComplexVectorProduct is a function that returns the product of an intrinsic
integer scalar and a MQC vector.
```

Parameters

in	*CompIn*	`CompIn is Complex(kind=real64)` The intrinsic complex to multiply.
in	*Vector*	`Vector is Type(MQC_Vector)` The MQC_Vector to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.3 mqc_integerscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_integerscalarmultiply (
            integer(kind=int64), intent(in) IntegerIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

```
MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by
an MQC_Scalar.
```

Parameters

in	*IntegerIn*	`IntegerIn is Integer(kind=int64)` The intrinsic integer variable to multiply.
in	*Scalar*	`Scalar is Type(MQC_Scalar)` The MQC_Scalar variable to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.4 mqc_integervectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_integervectorproduct (
            integer(kind=int64), intent(in) intIn,
            type(mqc_vector), intent(in) Vector )
```

MQC_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector

Purpose:

```
MQC_IntegerVectorProduct is a function that returns the product of an intrinsic
integer scalar and a MQC vector.
```

Parameters

in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer to multiply.`
in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`

Author

L. M. Thompson

Date

2019

6.44.2.5 mqc_matrixmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::operator(*)::mqc_matrixmatrixproduct (
            type(mqc_matrix), intent(in) MA,
            type(mqc_matrix), intent(in) MB )
```

MQC_MatrixMatrixProduct is a function that computes the element- wise product of two MQC matrices

Purpose:

```
MQC_MatrixMatrixProduct is a function that computes the element-wise product of
two MQC matrices.
```

Parameters

in	*MA*	`MA is Type(MQC_Matrix)` `The first MQC matrix.`
in	*MB*	`MB is Type(MQC_Matrix)` `The second MQC matrix.`

Author

H. P. Hratchian

Date

2017

6.44.2.6 mqc_matrixscalarproduct()

```
type(mqc_matrix) function mqc_algebra::operator(*)::mqc_matrixscalarproduct (
            type(mqc_matrix), intent(in) Matrix,
            type(mqc_scalar), intent(in) Scalar )
```

6.44.2.7 mqc_realscalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_realscalarmultiply (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

```
MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by
an MQC_Scalar.
```

Parameters

in	*realIn*	RealIn is Real(kind=real64) The intrinsic real variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.8 mqc_realvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_realvectorproduct (
            real(kind=real64), intent(in) RealIn,
            type(mqc_vector), intent(in) Vector )
```

MQC_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector

Purpose:

```
MQC_RealVectorProduct is a function that returns the product of an intrinsic
real scalar and a MQC vector.
```

Parameters

in	*RealIn*	RealIn is Real(kind=real64) The real intrinsic to multiply.
in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.9 mqc_scalarcomplexmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarcomplexmultiply (
            type(mqc_scalar), intent(in) Scalar,
            complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar

Purpose:

```
MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by
an MQC_Scalar.
```

Parameters

in	*ComplexIn*	Complex is Complex(kind=real64) The intrinsic complex variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.10 mqc_scalarintegermultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarintegermultiply (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

```
MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by
an MQC_Scalar.
```

Parameters

in	*IntegerIn*	IntegerIn is Integer(kind=int64) The intrinsic integer variable to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar varibale to multiply.

Author

L. M. Thompson

Date

2019

6.44.2.11 mqc_scalarmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::operator(*)::mqc_scalarmatrixproduct (
            type(mqc_scalar), intent(in) Scalar,
            type(mqc_matrix), intent(in) Matrix )
```

6.44.2.12 mqc_scalarmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarmultiply (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects

Purpose:

```
MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar to be multiplied.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar to be multiplied.`

Author

L. M. Thompson

Date

2016

6.44.2.13 mqc_scalarrealmultiply()

```
type(mqc_scalar) function mqc_algebra::operator(*)::mqc_scalarrealmultiply (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar

Purpose:

```
MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by
an MQC_Scalar.
```

Parameters

in	*realIn*	`RealIn is Real(kind=real64)` `The intrinsic real variable to multiply.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to multiply.`

Author

L. M. Thompson

Date

2019

6.44.2.14 mqc_scalarvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_scalarvectorproduct (
            type(mqc_scalar), intent(in) Scalar,
            type(mqc_vector), intent(in) Vector )
```

MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector

Purpose:

```
MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar
with a MQC vector.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.
in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to multiply.

Author

X. Sheng

A. D. Mahler

Date

2017, 2019

6.44.2.15 mqc_vectorcomplexproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorcomplexproduct (
            type(mqc_vector), intent(in) vector,
            complex(kind=real64), intent(in) compIn )
```

MQC_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar

Purpose:

```
MQC_VectorComplexProduct is a function that returns the product of a MQC vector
and an intrinsic complex scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`
in	*Comp← In*	`CompIn is Complex(kind=real64)` `The integer complex to multiply.`

Author

L. M. Thompson

Date

2019

6.44.2.16 mqc_vectorintegerproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorintegerproduct (
            type(mqc_vector), intent(in) vector,
            integer(kind=int64), intent(in) intIn )
```

MQC_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar

Purpose:

```
MQC_VectorIntegerProduct is a function that returns the product of a MQC vector
and an intrinsic integer scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`
in	*IntIn*	`IntIn is Integer(kind=int64)` `The integer intrinsic to multiply.`

Author

L. M. Thompson

Date

2019

6.44.2.17 mqc_vectorrealproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorrealproduct (
            type(mqc_vector), intent(in) vector,
            real(kind=real64), intent(in) realIn )
```

MQC_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar

Purpose:

```
MQC_VectorRealProduct is a function that returns the product of a MQC vector
and an intrinsic real scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to multiply.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The real intrinsic to multiply.`

Author

L. M. Thompson

Date

2019

6.44.2.18 mqc_vectorscalarproduct()

```
type(mqc_vector) function mqc_algebra::operator(*)::mqc_vectorscalarproduct (
            type(mqc_vector), intent(in) vector,
            type(mqc_scalar), intent(in) scalar )
```

MQC_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar

Purpose:

```
MQC_VectorScalarProduct is a function that returns the product of a MQC vector
with a MQC scalar.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to multiply.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar to multiply.

Author

X. Sheng

Date

2017

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.45 mqc_est::operator(∗) Interface Reference

Public Member Functions

- type(mqc_scf_integral) function mqc_scalar_integral_multiply (scalar, integral)
- type(mqc_scf_integral) function mqc_integral_scalar_multiply (integral, scalar)

6.45.1 Member Function/Subroutine Documentation

6.45.1.1 mqc_integral_scalar_multiply()

```
type(mqc_scf_integral) function mqc_est::operator(*)::mqc_integral_scalar_multiply (
            type(mqc_scf_integral), intent(in) integral,
            type(mqc_scalar), intent(in) scalar )
```

6.45.1.2 mqc_scalar_integral_multiply()

```
type(mqc_scf_integral) function mqc_est::operator(*)::mqc_scalar_integral_multiply (
            type(mqc_scalar), intent(in) scalar,
            type(mqc_scf_integral), intent(in) integral )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.46 mqc_algebra::operator(∗∗) Interface Reference

Exponentials a variable to the power of another

Public Member Functions

- type(mqc_scalar) function mqc_scalarexponent (Scalar1, Scalar2)

  MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar
- type(mqc_scalar) function mqc_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, CompIn)

  MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex

6.46.1 Detailed Description

Exponentials a variable to the power of another

6.46.2 Member Function/Subroutine Documentation

6.46.2.1 mqc_scalarcomplexexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarcomplexexponent (
            type(mqc_scalar), intent(in) Scalar,
            complex(kind=real64), intent(in) CompIn )
```

MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an intrinsic complex

Purpose:

```
MQC_ScalarComplexExponent is a function that raises an MQC_Scalar to the power of an
intrinsic complex.
```

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar)` `The base value.`
in	*Comp↩ In*	`CompIn is Complex(kind=real64)` `The power value.`

Author

L. M. Thompson

Date

2019

6.46.2.2 mqc_scalarexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarexponent (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar

Purpose:

```
MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of
another MQC_Scalar.
```

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar) The base value.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar) The power value.`

Author

L. M. Thompson

Date

2016

6.46.2.3 mqc_scalarintegerexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarintegerexponent (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an intrinsic integer

Purpose:

```
MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an
intrinsic integer.
```

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar) The base value.`
in	*IntIn*	`IntIn is Integer(kind=int64) The power value.`

Author

L. M. Thompson

Date

2019

6.46.2.4 mqc_scalarrealexponent()

```
type(mqc_scalar) function mqc_algebra::operator(**)::mqc_scalarrealexponent (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an intrinsic real

Purpose:

```
MQC_ScalarRealExponent is a function that raises an MQC_Scalar to the power of an
intrinsic real.
```

Parameters

in	*Scalar*	`Scalar1 is Type(MQC_Scalar)` `The base value.`
in	*Real↩ In*	`RealIn is Real(kind=real64)` `The power value.`

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.47 mqc_algebra::operator(+) Interface Reference

Sums two variables

Public Member Functions

- type(mqc_scalar) function mqc_scalaradd (Scalar1, Scalar2)
  MQC_ScalarAdd is a function that sums two MQC_Scalar objects
- type(mqc_scalar) function mqc_integerscalaradd (IntegerIn, Scalar)
  MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegeradd (Scalar, IntegerIn)
  MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_realscalaradd (RealIn, Scalar)
  MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealadd (Scalar, RealIn)
  MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_complexscalaradd (ComplexIn, Scalar)
  MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexadd (Scalar, ComplexIn)
  MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_vector) function mqc_vectorvectorsum (Vector1In, Vector2In)
  MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_scalarvectorsum (ScalarIn, VectorIn)
  MQC_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector
- type(mqc_matrix) function mqc_matrixmatrixsum (MA, MB)
  MQC_MatrixMatrixSum is a function that sums two MQC matrices

6.47.1 Detailed Description

Sums two variables

6.47.2 Member Function/Subroutine Documentation

6.47.2.1 mqc_complexscalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_complexscalaradd (
            complex(kind=real64), intent(in) ComplexIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

```
MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by
an MQC_Scalar.
```

Parameters

in	*Complex↔ In*	`Complex is Complex(kind=real64)` `The intrinsic complex variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variabel to sum.`

Author

L. M. Thompson

Date

2019

6.47.2.2 mqc_integerscalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_integerscalaradd (
            integer(kind=int64), intent(in) IntegerIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar

Purpose:

```
MQC_IntegerScalarAdd is a function that is used to sum an intrinsic integer by
an MQC_Scalar.
```

Parameters

in	*Integer↔ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to sum.`

Author

L. M. Thompson

Date

2019

6.47.2.3 mqc_matrixmatrixsum()

```
type(mqc_matrix) function mqc_algebra::operator(+)::mqc_matrixmatrixsum (
            type(mqc_matrix), intent(in) MA,
            type(mqc_matrix), intent(in) MB )
```

MQC_MatrixMatrixSum is a function that sums two MQC matrices

Purpose:

```
MQC_MatrixMatrixSum is a function that sums two MQC matrices.
```

Parameters

in	*MA*	MA is Type(MQC_Matrix) First MQC matrix to sum.
in	*MB*	MB is Type(MQC_Matrix) Second MQC matrix to sum.

Author

H. P. Hratchian

L. M. Thompson

Date

2017, 2018

6.47.2.4 mqc_realscalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_realscalaradd (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

```
MQC_RealScalarAdd is a function that is used to sum an intrinsic real by
an MQC_Scalar.
```

Parameters

in	*realIn*	`RealIn is Real(kind=real64)` `The intrinsic real variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to sum.`

Author

L. M. Thompson

Date

2019

6.47.2.5 mqc_scalaradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalaradd (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarAdd is a function that sums two MQC_Scalar objects

Purpose:

```
MQC_ScalarAdd is a function that sums two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar to be summed.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar to be summed.`

Author

L. M. Thompson

Date

2016

6.47.2.6 mqc_scalarcomplexadd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarcomplexadd (
            type(mqc_scalar), intent(in) Scalar,
            complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar

Purpose:

```
MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by
an MQC_Scalar.
```

Parameters

in	*ComplexIn*	Complex is Complex(kind=real64) The intrinsic complex variable to sum.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variabel to sum.

Author

L. M. Thompson

Date

2019

6.47.2.7 mqc_scalarintegeradd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarintegeradd (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar

Purpose:

```
MQC_ScalarIntegerSum is a function that is used to sum an intrinsic integer by
an MQC_Scalar.
```

Parameters

in	*Integer←↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar varibale to sum.`

Author

L. M. Thompson

Date

2019

6.47.2.8 mqc_scalarrealadd()

```
type(mqc_scalar) function mqc_algebra::operator(+)::mqc_scalarrealadd (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar

Purpose:

```
MQC_ScalarRealSum is a function that is used to sum an intrinsic real by
an MQC_Scalar.
```

Parameters

in	*realIn*	`RealIn is Real(kind=real64)` `The intrinsic real variable to sum.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to sum.`

Author

L. M. Thompson

Date

2019

6.47.2.9 mqc_scalarvectorsum()

```
type(mqc_vector) function mqc_algebra::operator(+)::mqc_scalarvectorsum (
            type(mqc_scalar), intent(in) ScalarIn,
            type(mqc_vector), intent(in) VectorIn )
```

MQC_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector

Purpose:

```
MQC_VectorVectorSum is a function that adds an MQC scalar to all elements of
an MQC vector.
```

Parameters

in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The MQC scalar to add to the MQC vector.
in	*VectorIn*	VectorIn is Type(MQC_Vector) The MQC vector with elements to sum with ScalarIn.

Author

L. M. Thompson

Date

2016

6.47.2.10 mqc_vectorvectorsum()

```
type(mqc_vector) function mqc_algebra::operator(+)::mqc_vectorvectorsum (
            type(mqc_vector), intent(in) Vector1In,
            type(mqc_vector), intent(in) Vector2In )
```

MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector

Purpose:

```
MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in
another MQC vector.
```

Parameters

in	*Vector1↩ In*	Vector1In is Type(MQC_Vector) The first MQC vector that will be summed.
in	*Vector2↩ In*	Vector2In is Type(MQC_Vector) The second MQC vector that will be summed.

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.48 mqc_est::operator(+) Interface Reference

Public Member Functions

- type(mqc_scf_integral) function mqc_integral_sum (integralA, integralB)

6.48.1 Member Function/Subroutine Documentation

6.48.1.1 mqc_integral_sum()

```
type(mqc_scf_integral) function mqc_est::operator(+)::mqc_integral_sum (
            type(mqc_scf_integral), intent(in) integralA,
            type(mqc_scf_integral), intent(in) integralB )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.49 mqc_est::operator(-) Interface Reference

Public Member Functions

- type(mqc_scf_integral) function mqc_integral_difference (integralA, integralB)

6.49.1 Member Function/Subroutine Documentation

6.49.1.1 mqc_integral_difference()

```
type(mqc_scf_integral) function mqc_est::operator(-)::mqc_integral_difference (
            type(mqc_scf_integral), intent(in) integralA,
            type(mqc_scf_integral), intent(in) integralB )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.50 mqc_algebra::operator(-) Interface Reference

Subtracts two variables

Public Member Functions

- type(mqc_scalar) function mqc_scalarsubtract (Scalar1, Scalar2)
  MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects
- type(mqc_scalar) function mqc_integerscalarsubtract (IntegerIn, Scalar)
  MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic integer
- type(mqc_scalar) function mqc_scalarintegersubtract (Scalar, IntegerIn)
  MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar
- type(mqc_scalar) function mqc_realscalarsubtract (RealIn, Scalar)
  MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic real
- type(mqc_scalar) function mqc_scalarrealsubtract (Scalar, RealIn)
  MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar
- type(mqc_scalar) function mqc_complexscalarsubtract (ComplexIn, Scalar)
  MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic complex
- type(mqc_scalar) function mqc_scalarcomplexsubtract (Scalar, ComplexIn)
  MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar
- type(mqc_vector) function mqc_vectorvectordifference (Vector1In, Vector2In)
  MQC_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_scalarvectordifference (ScalarIn, VectorIn)
  MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector
- type(mqc_matrix) function mqc_matrixmatrixsubtract (MA, MB)
  MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices

6.50.1 Detailed Description

Subtracts two variables

6.50.2 Member Function/Subroutine Documentation

6.50.2.1 mqc_complexscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_complexscalarsubtract (
            complex(kind=real64), intent(in) ComplexIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic complex

Purpose:

```
MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an
intrinisic complex.
```

Parameters

in	*ComplexIn*	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract from.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract.

Author

L. M. Thompson

Date

2019

6.50.2.2 mqc_integerscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_integerscalarsubtract (
            integer(kind=int64), intent(in) IntegerIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic integer

Purpose:

```
MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an
intrinisic integer.
```

Parameters

in	*Integer↵ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer to subtract from.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract.`

Author

L. M. Thompson

Date

2019

6.50.2.3 mqc_matrixmatrixsubtract()

```
type(mqc_matrix) function mqc_algebra::operator(-)::mqc_matrixmatrixsubtract (
            type(mqc_matrix), intent(in) MA,
            type(mqc_matrix), intent(in) MB )
```

MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices

Purpose:

```
MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices.
```

Parameters

in	*MA*	`MA is Type(MQC_Matrix)` `The matrix that MB will be subtracted from.`
in	*MB*	`MB is Type(MQC_Matrix)` `The matrix that will be subtracted from MA.`

Author

H. P. Hratchian

Date

2017

6.50.2.4 mqc_realscalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_realscalarsubtract (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic real

Purpose:

```
MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an
intrinisic real.
```

Parameters

in	*RealIn*	`RealIn is Real(kind=real64)` `The intrinsic real to subtract from.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable to subtract.`

Author

L. M. Thompson

Date

2019

6.50.2.5 mqc_scalarcomplexsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarcomplexsubtract (
            type(mqc_scalar), intent(in) Scalar,
            complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar

Purpose:

```
MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex
from an MQC_Scalar.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from.
in	*ComplexIn*	ComplexIn is Complex(kind=real64) The intrinsic complex to subtract.

Author

L. M. Thompson

Date

2019

6.50.2.6 mqc_scalarintegersubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarintegersubtract (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar

Purpose:

```
MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer
from an MQC_Scalar.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from.
in	*IntegerIn*	IntegerIn is Integer(kind=int64) The intrinsic integer to subtract.

Author

L. M. Thompson

Date

2019

6.50.2.7 mqc_scalarrealsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarrealsubtract (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar

Purpose:

```
MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an
MQC_Scalar.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable to subtract from.
in	*Real↩ In*	RealIn is Real(kind=real64) The intrinsic real to subtract.

Author

L. M. Thompson

Date

2019

6.50.2.8 mqc_scalarsubtract()

```
type(mqc_scalar) function mqc_algebra::operator(-)::mqc_scalarsubtract (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects

Purpose:

```
MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects.
```

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar from which Scalar2 will be subtracted.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar which will be subtracted from Scalar1.`

Author

L. M. Thompson

Date

2016

6.50.2.9 mqc_scalarvectordifference()

```
type(mqc_vector) function mqc_algebra::operator(-)::mqc_scalarvectordifference (
            type(mqc_scalar), intent(in) ScalarIn,
            type(mqc_vector), intent(in) VectorIn )
```

MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector

Purpose:

```
MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from
all elements of an MQC vector.
```

Parameters

in	*Scalar↩ In*	`ScalarIn is Type(MQC_Scalar)` `The MQC scalar to be subtracted from elements of the the MQC vector.`
in	*Vector↩ In*	`VectorIn is Type(MQC_Vector)` `The MQC vector with elements from which ScalarIn will be subtracted.`

Author

L. M. Thompson

Date

2016

6.50.2.10 mqc_vectorvectordifference()

```
type(mqc_vector) function mqc_algebra::operator(-)::mqc_vectorvectordifference (
            type(mqc_vector), intent(in) Vector1In,
            type(mqc_vector), intent(in) Vector2In )
```

MQC_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector

Purpose:

```
MQC_VectorVectorDifference is a function that subtracts two MQC vectors and
stores them in another MQC vector.
```

Parameters

in	*Vector1↩ In*	`Vector1In is Type(MQC_Vector)` `The first MQC vector from which the second will be subtracted.`
in	*Vector2↩ In*	`Vector2In is Type(MQC_Vector)` `The second MQC vector that will be subtracted from the first.`

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.51 mqc_algebra::operator(.dot.) Interface Reference

Computes the inner product of two arrays

Public Member Functions

- type(mqc_scalar) function mqc_vectorvectordotproduct (Vector1, Vector2)

  MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors
- type(mqc_vector) function mqc_vectormatrixdotproduct (VA, MB)
- type(mqc_vector) function mqc_matrixvectordotproduct (MA, VB)
- type(mqc_matrix) function mqc_matrixmatrixdotproduct (MA, MB)

6.51.1 Detailed Description

Computes the inner product of two arrays

6.51.2 Member Function/Subroutine Documentation

6.51.2.1 mqc_matrixmatrixdotproduct()

```
type(mqc_matrix) function mqc_algebra::operator(.dot.)::mqc_matrixmatrixdotproduct (
            type(mqc_matrix), intent(in) MA,
            type(mqc_matrix), intent(in) MB )
```

6.51.2.2 mqc_matrixvectordotproduct()

```
type(mqc_vector) function mqc_algebra::operator(.dot.)::mqc_matrixvectordotproduct (
            type(mqc_matrix), intent(in) MA,
            type(mqc_vector), intent(in) VB )
```

6.51.2.3 mqc_vectormatrixdotproduct()

```
type(mqc_vector) function mqc_algebra::operator(.dot.)::mqc_vectormatrixdotproduct (
            type(mqc_vector), intent(in) VA,
            type(mqc_matrix), intent(in) MB )
```

6.51.2.4 mqc_vectorvectordotproduct()

```
type(mqc_scalar) function mqc_algebra::operator(.dot.)::mqc_vectorvectordotproduct (
            type(mqc_vector), intent(in) Vector1,
            type(mqc_vector), intent(in) Vector2 )
```

MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors

Purpose:

```
MQC_VectorVectorDotProduct is a function that returns the dot product of two
MQC vectors. The first vector should be a row vector, while the second vector
should be a column vector. The vectors should be of the same length.
```

Parameters

in	*Vector1*	Vector1 is Type(MQC_Vector) The MQC row vector.
in	*Vector2*	Vector2 is Type(MQC_Vector) The MQC column vector.

Author

H. P. Hratchian

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.52 mqc_algebra::operator(.eq.) Interface Reference

Determines if two variables are equal

Public Member Functions

- logical function mqc_scalareq (Scalar1, Scalar2)
  MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal

6.52.1 Detailed Description

Determines if two variables are equal

6.52.2 Member Function/Subroutine Documentation

6.52.2.1 mqc_scalareq()

```
logical function mqc_algebra::operator(.eq.)::mqc_scalareq (
        type(mqc_scalar), intent(in) Scalar1,
        type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables are equal

Purpose:

```
MQC_ScalarEQ is a function that returns TRUE if two MQC_Scalar variables
are equal.
```

Parameters

in	*Scalar1*	`Scalar1 is Type(MQC_Scalar)` `The first MQC_Scalar that will be tested.`
in	*Scalar2*	`Scalar2 is Type(MQC_Scalar)` `The second MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.53 mqc_algebra::operator(.ewd.) Interface Reference

Computes the element-wise quotient of two arrays

Public Member Functions

- type(mqc_matrix) function mqc_elementmatrixdivide (A, B)
    MQC_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices

6.53.1 Detailed Description

Computes the element-wise quotient of two arrays

6.53.2 Member Function/Subroutine Documentation

6.53.2.1 mqc_elementmatrixdivide()

```
type(mqc_matrix) function mqc_algebra::operator(.ewd.)::mqc_elementmatrixdivide (
            type(mqc_matrix), intent(in) A,
            type(mqc_matrix), intent(in) B )
```

MQC_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices

Purpose:

```
MQC_ElementMatrixDivide is a function that returns the element-wise quotient
of two MQC matrices.
```

Parameters

in	*A*	`A is type(mqc_matrix)` `The matrix with elements being the numerator.`
in	*B*	`B is type(mqc_matrix)` `The matrix with elements being the denominator.`

Author

X. Sheng

Date

2017

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.54 mqc_algebra::operator(.ewp.) Interface Reference

Computes the element-wise product of two arrays

Public Member Functions

- type(mqc_vector) function mqc_elementvectorproduct (Vector1In, Vector2In)

  MQC_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector
- type(mqc_matrix) function mqc_elementmatrixproduct (A, B)

  MQC_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices

6.54.1 Detailed Description

Computes the element-wise product of two arrays

6.54.2 Member Function/Subroutine Documentation

6.54.2.1 mqc_elementmatrixproduct()

```
type(mqc_matrix) function mqc_algebra::operator(.ewp.)::mqc_elementmatrixproduct (
            type(mqc_matrix), intent(in) A,
            type(mqc_matrix), intent(in) B )
```

MQC_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices

Purpose:

```
MQC_ElementMatrixProduct is a function that returns the element-wise product
of two MQC matrices.
```

Parameters

in	*A*	A is type(mqc_matrix) The first matrix to element-wise multiply.
in	*B*	B is type(mqc_matrix) The second matrix to element-wise multiply.

Author

X. Sheng

Date

2017

6.54.2.2 mqc_elementvectorproduct()

```
type(mqc_vector) function mqc_algebra::operator(.ewp.)::mqc_elementvectorproduct (
            type(mqc_vector), intent(in) Vector1In,
            type(mqc_vector), intent(in) Vector2In )
```

MQC_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector

Purpose:

```
MQC_ElementVectorProduct is a function that multiplies two MQC vectors
elementwise and stores them into another MQC vector.
```

Parameters

in	*Vector1↩ In*	`Vector1In is Type(MQC_Vector)` `The frist MQC vector to multiply elementwise.`
in	*Vector2↩ In*	`Vector2In is Type(MQC_Vector)` `The second MQC vector to multiply elementwise.`

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.55 mqc_algebra::operator(.ge.) Interface Reference

Determines if a variable is greater than or equal to another

Public Member Functions

- logical function mqc_scalarge (Scalar1, Scalar2)

  MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_↩ Scalar

6.55.1 Detailed Description

Determines if a variable is greater than or equal to another

6.55.2 Member Function/Subroutine Documentation

6.55.2.1 mqc_scalarge()

```
logical function mqc_algebra::operator(.ge.)::mqc_scalarge (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC←_Scalar

Purpose:

```
MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or
equal to the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part is
is greater than or equal to the right real part and FALSE if the left real part is less
than the right real part.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.56 mqc_algebra::operator(.gt.) Interface Reference

Determines if a variable is greater than another

Public Member Functions

- logical function mqc_scalargt (Scalar1, Scalar2)

  MQC_ScalarGT is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar
- logical function mqc_scalargtinteger (Scalar, IntIn)

  MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer
- logical function mqc_integergtscalar (IntIn, Scalar)

  MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar
- logical function mqc_scalargtreal (Scalar, RealIn)

  MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real
- logical function mqc_realgtscalar (RealIn, Scalar)

  MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar

6.56.1 Detailed Description

Determines if a variable is greater than another

6.56.2 Member Function/Subroutine Documentation

6.56.2.1 mqc_integergtscalar()

```
logical function mqc_algebra::operator(.gt.)::mqc_integergtscalar (
            integer(kind=int64), intent(in) IntIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater than a MQC_Scalar

Purpose:

```
MQC_IntegerGTScalar is a function that returns TRUE if an intrinsic integer is greater
than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer
is greater than the real part of the MQC_Scalar and FALSE if the intrinsic integer is
less than the real part of the MQC_Scalar. If the intrinsic integer is equal to the
real part of the MQC_Scalar, the function returns TRUE if the imaginary part of
MQC_Scalar is less than zero and FALSE otherwise.
```

Parameters

in	*IntIn*	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.56.2.2 mqc_realgtscalar()

```
logical function mqc_algebra::operator(.gt.)::mqc_realgtscalar (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar

Purpose:

```
MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater
than a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real
is greater than the real part of the MQC_Scalar and FALSE if the intrinsic real is
less than the real part of the MQC_Scalar. If the intrinsic real is equal to the
real part of the MQC_Scalar, the function returns TRUE if the imaginary part of
MQC_Scalar is less than zero and FALSE otherwise.
```

Parameters

in	*RealIn*	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.56.2.3 mqc_scalargt()

```
logical function mqc_algebra::operator(.gt.)::mqc_scalargt (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarGT is a function that returns TRUE if the left MQC_Scalar is greater than the right MQC_Scalar

Purpose:

```
MQC_ScalarGT is a function that returns TRUE if the left MQC_Scalar is greater than
the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part
is greater than the right real part and FALSE if the left real part is less than
the right real part. If the left real part is equal to the right real part, the
function returns TRUE if the left imaginary part is greater than the right imaginary
part and FALSE otherwise.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.56.2.4 mqc_scalargtinteger()

```
logical function mqc_algebra::operator(.gt.)::mqc_scalargtinteger (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic integer

Purpose:

```
MQC_ScalarGTInteger is a function that returns TRUE if a MQC_Scalar is greater than an
intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the
MQC_Scalar is greater than the intrinsic integer and FALSE if the real part of the
MQC_Scalar is less than the intrinsic integer. If the real part of the MQC_Scalar is
equal to the intrinsic integer, the function returns TRUE if the imaginary part of
MQC_Scalar is greater than zero and FALSE otherwise.
```

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`
in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer that will be tested.`

Author

L. M. Thompson

Date

2019

6.56.2.5 mqc_scalargtreal()

```
logical function mqc_algebra::operator(.gt.)::mqc_scalargtreal (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real

Purpose:

```
MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an
intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the
MQC_Scalar is greater than the intrinsic real and FALSE if the real part of the
MQC_Scalar is less than the intrinsic real. If the real part of the MQC_Scalar is
equal to the intrinsic real, the function returns TRUE if the imaginary part of
MQC_Scalar is greater than zero and FALSE otherwise.
```

Parameters

in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`
in	*Real↔ In*	`RealIn is Real(kind=int64)` `The intrinsic real that will be tested.`

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.57 mqc_algebra::operator(.le.) Interface Reference

Determines if a variable is less than or equal to another

Public Member Functions

- logical function mqc_scalarle (Scalar1, Scalar2)
  MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar
- logical function mqc_scalarlereal (Scalar, RealIn)
  MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real
- logical function mqc_reallescalar (RealIn, Scalar)
  MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar
- logical function mqc_scalarleinteger (Scalar, IntIn)
  MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer
- logical function mqc_integerlescalar (IntIn, Scalar)
  MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar

6.57.1 Detailed Description

Determines if a variable is less than or equal to another

6.57.2 Member Function/Subroutine Documentation

6.57.2.1 mqc_integerlescalar()

```
logical function mqc_algebra::operator(.le.)::mqc_integerlescalar (
            integer(kind=int64), intent(in) IntIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_↩ Scalar

Purpose:

```
MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less
than or equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic integer
is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic
integer is greater than the real part of the MQC_Scalar.
```

Parameters

in	*IntIn*	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.57.2.2 mqc_reallescalar()

```
logical function mqc_algebra::operator(.le.)::mqc_reallescalar (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar

Purpose:

```
MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or
equal to a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real
is less than or equal to the real part of the MQC_Scalar and FALSE if the intrinsic
real is greater than the real part of the MQC_Scalar.
```

Parameters

in	*Real↵In*	`RealIn is Real(kind=real64)` `The intrinsic real that will be tested.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar that will be tested.`

Author

L. M. Thompson

Date

2019

6.57.2.3 mqc_scalarle()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarle (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar

Purpose:

```
MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or
equal to the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part
is less than or equal to the right real part and FALSE if the left real part is greater
than the right real part.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.57.2.4 mqc_scalarleinteger()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarleinteger (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntIn )
```

MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer

Purpose:

```
MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or
equal to an intrinsic integer.

When dealing with complex numbers, the function returns TRUE if the real part of the
MQC_Scalar is less than or equal to the intrinsic integer and FALSE if the real part
of the MQC_Scalar is greater than the intrinsic integer.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	*IntIn*	IntIn is Integer(kind=int64) The intrinsic integer that will be tested.

Author

L. M. Thompson

Date

2019

6.57.2.5 mqc_scalarlereal()

```
logical function mqc_algebra::operator(.le.)::mqc_scalarlereal (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real

Purpose:

```
MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to
an intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the
MQC_Scalar is less than or equal to the intrinsic real and FALSE if the real part of the
MQC_Scalar is greater than the intrinsic real.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	*Real↩ In*	RealIn is Real(kind=int64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.58 mqc_algebra::operator(.lt.) Interface Reference

Determines if a variable is less than another

Public Member Functions

- logical function mqc_scalarlt (Scalar1, Scalar2)

  MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar
- logical function mqc_scalarltreal (Scalar, RealIn)

  MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real
- logical function mqc_realltscalar (RealIn, Scalar)

  MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

6.58.1 Detailed Description

Determines if a variable is less than another

6.58.2 Member Function/Subroutine Documentation

6.58.2.1 mqc_realltscalar()

```
logical function mqc_algebra::operator(.lt.)::mqc_realltscalar (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than a MQC_Scalar

Purpose:

```
MQC_RealLTScalar is a function that returns TRUE if an intrinsic real is less than
a MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the intrinsic real
is less than the real part of the MQC_Scalar and FALSE if the intrinsic real is
greater than the real part of the MQC_Scalar. If the intrinsic real is equal to the
real part of the MQC_Scalar, the function returns TRUE if the imaginary part of
MQC_Scalar is greater than zero and FALSE otherwise.
```

Parameters

in	*RealIn*	RealIn is Real(kind=real64) The intrinsic real that will be tested.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2019

6.58.2.2 mqc_scalarlt()

```
logical function mqc_algebra::operator(.lt.)::mqc_scalarlt (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than the right MQC_Scalar

Purpose:

```
MQC_ScalarLT is a function that returns TRUE if the left MQC_Scalar is less than
the right MQC_Scalar.

When dealing with complex numbers, the function returns TRUE if the left real part
is less than the right real part and FALSE if the left real part is greater than
the right real part. If the left real part is equal to the right real part, the
function returns TRUE if the left imaginary part is less than the right imaginary
part and FALSE otherwise.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

6.58.2.3 mqc_scalarltreal()

```
logical function mqc_algebra::operator(.lt.)::mqc_scalarltreal (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an intrinsic real

Purpose:

```
MQC_ScalarLTReal is a function that returns TRUE if a MQC_Scalar is less than an
intrinsic real.

When dealing with complex numbers, the function returns TRUE if the real part of the
MQC_Scalar is less than the intrinsic real and FALSE if the real part of the MQC_Scalar
is greater than the intrinsic real. If the real part of the MQC_Scalar is equal to the
intrinsic real, the function returns TRUE if the imaginary part of MQC_Scalar is less
than zero and FALSE otherwise.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar that will be tested.
in	*Real↔In*	RealIn is Real(kind=real64) The intrinsic real that will be tested.

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.59 mqc_algebra::operator(.ne.) Interface Reference

Determines if two variables are not equal

Public Member Functions

- logical function mqc_scalarne (Scalar1, Scalar2)

  MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal

6.59.1 Detailed Description

Determines if two variables are not equal

6.59.2 Member Function/Subroutine Documentation

6.59.2.1 mqc_scalarne()

```
logical function mqc_algebra::operator(.ne.)::mqc_scalarne (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables are not equal

Purpose:

```
MQC_ScalarNE is a function that returns TRUE if two MQC_Scalar variables
are not equal.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The first MQC_Scalar that will be tested.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The second MQC_Scalar that will be tested.

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.60 mqc_algebra::operator(.outer.) Interface Reference

Computes the outer product of two vectors

Public Member Functions

- type(mqc_matrix) function mqc_outer (VA, VB)

  MQC_Outer is a function that returns the outer product of two MQC vectors

6.60.1 Detailed Description

Computes the outer product of two vectors

6.60.2 Member Function/Subroutine Documentation

6.60.2.1 mqc_outer()

```
type(mqc_matrix) function mqc_algebra::operator(.outer.)::mqc_outer (
            type(mqc_vector), intent(in) VA,
            type(mqc_vector), intent(in) VB )
```

MQC_Outer is a function that returns the outer product of two MQC vectors

Purpose:

```
MQC_Outer is a function that returns the outer product of two MQC vectors. The
first vector should be a column vector, while the second vector should be a row
vector.
```

Parameters

in	*VA*	VA is Type(MQC_Vector) The MQC column vector.
in	*VB*	VB is Type(MQC_Vector) The MQC row vector.

Author

X. Sheng

Date

2017

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.61 mqc_algebra::operator(.x.) Interface Reference

Computes the cross product of two vectors

Public Member Functions

- type(mqc_vector) function mqc_crossproduct (Vector1In, Vector2In)
  MQC_CrossProduct is a function that returns the cross product of two MQC vectors

6.61.1 Detailed Description

Computes the cross product of two vectors

6.61.2 Member Function/Subroutine Documentation

6.61.2.1 mqc_crossproduct()

```
type(mqc_vector) function mqc_algebra::operator(.x.)::mqc_crossproduct (
            type(mqc_vector), intent(in) Vector1In,
            type(mqc_vector), intent(in) Vector2In )
```

MQC_CrossProduct is a function that returns the cross product of two MQC vectors

Purpose:

```
MQC_CrossProduct is a function that returns the cross product of two MQC
vectors. The vectors should both be of length 3.
```

Parameters

in	*Vector1←In*	`Vector1In is Type(MQC_Vector) The first MQC vector.`
in	*Vector2←In*	`Vector2In is Type(MQC_Vector) The second MQC vector.`

Author

L. M. Thompson

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.62 mqc_algebra::operator(/) Interface Reference

Divides two variables

Public Member Functions

- type(mqc_scalar) function mqc_scalardivide (Scalar1, Scalar2)
  MQC_ScalarDivide is a function that divides two MQC_Scalar objects
- type(mqc_scalar) function mqc_integerscalardivide (IntegerIn, Scalar)
  MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarintegerdivide (Scalar, IntegerIn)
  MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer
- type(mqc_scalar) function mqc_realscalardivide (RealIn, Scalar)
  MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarrealdivide (Scalar, RealIn)
  MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an intrinsic real
- type(mqc_scalar) function mqc_complexscalardivide (ComplexIn, Scalar)
  MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_scalarcomplexdivide (Scalar, ComplexIn)
  MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex
- type(mqc_vector) function mqc_vectorscalardivide (vector, scalar)
  MQC_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar
- type(mqc_vector) function mqc_vectorrealdivide (vector, realIn)
  MQC_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer
- type(mqc_vector) function mqc_vectorintegerdivide (vector, intIn)
  MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar
- type(mqc_vector) function mqc_vectorcomplexdivide (vector, compIn)
  MQC_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar

6.62.1 Detailed Description

Divides two variables

6.62.2 Member Function/Subroutine Documentation

6.62.2.1 mqc_complexscalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_complexscalardivide (
            complex(kind=real64), intent(in) ComplexIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar

Purpose:

MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar.

Parameters

in	*ComplexIn*	ComplexIn is Complex(kind=real64) The intrinsic complex variable numerator.
in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable denominator.

Author

L. M. Thompson

Date

2019

6.62.2.2 mqc_integerscalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_integerscalardivide (
            integer(kind=int64), intent(in) IntegerIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar

Purpose:

```
MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by
an MQC_Scalar.
```

Parameters

in	*Integer↩ In*	`IntegerIn is Integer(kind=int64)` `The intrinsic integer variable numerator.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable denominator.`

Author

L. M. Thompson

Date

2019

6.62.2.3 mqc_realscalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_realscalardivide (
            real(kind=real64), intent(in) RealIn,
            type(mqc_scalar), intent(in) Scalar )
```

MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar

Purpose:

```
MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an
MQC_Scalar.
```

Parameters

in	*Real↩ In*	`RealIn is Real(kind=real64)` `The intrinsic real variable numerator.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar variable denominator.`

Author

L. M. Thompson

Date

2019

6.62.2.4 mqc_scalarcomplexdivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarcomplexdivide (
            type(mqc_scalar), intent(in) Scalar,
            complex(kind=real64), intent(in) ComplexIn )
```

MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex

Purpose:

```
MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an
intrinsic complex.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	*Complex↩ In*	ComplexIn is Complex(kind=real64) The intrinsic complex variable denominator.

Author

L. M. Thompson

Date

2019

6.62.2.5 mqc_scalardivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalardivide (
            type(mqc_scalar), intent(in) Scalar1,
            type(mqc_scalar), intent(in) Scalar2 )
```

MQC_ScalarDivide is a function that divides two MQC_Scalar objects

Purpose:

```
MQC_ScalarDivide is a function that divides MQC_Scalar objects.
```

Parameters

in	*Scalar1*	Scalar1 is Type(MQC_Scalar) The numerator.
in	*Scalar2*	Scalar2 is Type(MQC_Scalar) The denominator.

Author

L. M. Thompson

Date

2016

6.62.2.6 mqc_scalarintegerdivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarintegerdivide (
            type(mqc_scalar), intent(in) Scalar,
            integer(kind=int64), intent(in) IntegerIn )
```

MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer

Purpose:

```
MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an
intrinsic integer.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	*Integer↩ In*	IntegerIn is Integer(kind=int64) The intrinsic integer variable denominator.

Author

L. M. Thompson

Date

2019

6.62.2.7 mqc_scalarrealdivide()

```
type(mqc_scalar) function mqc_algebra::operator(/)::mqc_scalarrealdivide (
            type(mqc_scalar), intent(in) Scalar,
            real(kind=real64), intent(in) RealIn )
```

MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an intrinsic real

Purpose:

```
MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an
intrinsic real.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The MQC_Scalar variable numerator.
in	*Real↩ In*	RealIn is Real(kind=real64) The intrinsic real variable denominator.

Author

L. M. Thompson

Date

2019

6.62.2.8 mqc_vectorcomplexdivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorcomplexdivide (
            type(mqc_vector), intent(in) vector,
            complex(kind=real64), intent(in) compIn )
```

MQC_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar

Purpose:

```
MQC_VectorComplexDivide is a function that returns a MQC vector divided by an
intrinsic complex scalar.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC_Vector to divide.
in	*Comp↵ In*	CompIn is Complex(kind=comp64) The intrinsic complex scalar to divide by.

Author

L. M. Thompson

Date

2019

6.62.2.9 mqc_vectorintegerdivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorintegerdivide (
            type(mqc_vector), intent(in) vector,
            integer(kind=int64), intent(in) intIn )
```

MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar

Purpose:

```
MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an
intrinsic integer scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to divide.`
in	*IntIn*	`IntIn is Integer(kind=int64)` `The intrinsic integer scalar to divide by.`

Author

L. M. Thompson

Date

2019

6.62.2.10 mqc_vectorrealdivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorrealdivide (
            type(mqc_vector), intent(in) vector,
            real(kind=real64), intent(in) realIn )
```

MQC_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer

Purpose:

```
MQC_VectorRealDivide is a function that returns a MQC vector divided by an
intrinsic real scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to divide.`
in	*Real↔ In*	`RealIn is Real(kind=real64)` `The intrinsic real scalar to divide by.`

Author

L. M. Thompson

Date

2019

6.62.2.11 mqc_vectorscalardivide()

```
type(mqc_vector) function mqc_algebra::operator(/)::mqc_vectorscalardivide (
            type(mqc_vector), intent(in) vector,
            type(mqc_scalar), intent(in) scalar )
```

MQC_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar

Purpose:

```
MQC_VectorScalarDivide is a function that returns a MQC vector divided by a
MQC scalar.
```

Parameters

in	*Vector*	`Vector is Type(MQC_Vector)` `The MQC_Vector to divide.`
in	*Scalar*	`Scalar is Type(MQC_Scalar)` `The MQC_Scalar to divide by.`

Author

A. D. Mahler

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.63 mqc_algebra::real Interface Reference

Returns the real part

Public Member Functions

- type(mqc_scalar) function mqc_scalar_complex_realpart (ScalarIn)

  MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar

- type(mqc_vector) function mqc_vector_complex_realpart (A)

  MQC_Vector_Complex_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

6.63.1 Detailed Description

Returns the real part

6.63.2 Member Function/Subroutine Documentation

6.63.2.1 mqc_scalar_complex_realpart()

```
type(mqc_scalar) function mqc_algebra::real::mqc_scalar_complex_realpart (
            type(mqc_scalar), intent(in) ScalarIn )
```

MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar

Purpose:

```
MQC_Scalar_Complex_RealPart is a function that returns the real part of an
MQC_Scalar.
```

Parameters

in	*ScalarIn*	ScalarIn is Type(MQC_Scalar) The MQC_Scalar input variable.

Author

L. M. Thompson

Date

2019

6.63.2.2 mqc_vector_complex_realpart()

```
type(mqc_vector) function mqc_algebra::real::mqc_vector_complex_realpart (
            class(mqc_vector), intent(in) A )
```

MQC_Vector_Complex_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector

Purpose:

```
MQC_Vector_Complex_RealPart is a function that returns a MQC vector with
elements containing the real part of elements of another MQC vector.
```

Parameters

in	*A*	A is Class(MQC_Vector) The name of the MQC_Vector variable.

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.64 mqc_algebra::sin Interface Reference

Returns the sine

Public Member Functions

- type(mqc_scalar) function mqc_scalar_sin (Scalar)
  MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

6.64.1 Detailed Description

Returns the sine

6.64.2 Member Function/Subroutine Documentation

6.64.2.1 mqc_scalar_sin()

```
type(mqc_scalar) function mqc_algebra::sin::mqc_scalar_sin (
            type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar

Purpose:

```
MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The argument of the function.

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.65 mqc_algebra::sqrt Interface Reference

Returns the square root

Public Member Functions

- type(mqc_scalar) function mqc_scalar_sqrt (Scalar)

  MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

6.65.1 Detailed Description

Returns the square root

6.65.2 Member Function/Subroutine Documentation

6.65.2.1 mqc_scalar_sqrt()

```
type(mqc_scalar) function mqc_algebra::sqrt::mqc_scalar_sqrt (
            type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar

Purpose:

```
MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar.
```

Parameters

in	*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

6.66 mqc_algebra::tan Interface Reference

Returns the tangent

Public Member Functions

- type(mqc_scalar) function mqc_scalar_tan (Scalar)

  MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar

6.66.1 Detailed Description

Returns the tangent

6.66.2 Member Function/Subroutine Documentation

6.66.2.1 mqc_scalar_tan()

```
type(mqc_scalar) function mqc_algebra::tan::mqc_scalar_tan (
            type(mqc_scalar), intent(in) Scalar )
```

MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar

Purpose:

```
MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar.
```

Parameters

in	*Scalar*	Scalar is Type(MQC_Scalar) The argument of the function.

Author

L. M. Thompson

Date

2019

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

6.67 mqc_est::transpose Interface Reference

Public Member Functions

- type(mqc_scf_integral) function mqc_integral_transpose (integral, label)

6.67.1 Member Function/Subroutine Documentation

6.67.1.1 mqc_integral_transpose()

```
type(mqc_scf_integral) function mqc_est::transpose::mqc_integral_transpose (
            type(mqc_scf_integral), intent(in) integral,
            character(len=*), intent(in), optional label )
```

The documentation for this interface was generated from the following file:

- src/mqc_est.F03

6.68 mqc_algebra::transpose Interface Reference

Returns the transpose

Public Member Functions

- type(mqc_vector) function mqc_vector_transpose (Vector)
  MQC_Vector_Transpose is a function that returns the transpose of an MQC vector
- type(mqc_matrix) function mqc_matrix_transpose (Matrix)
  MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix

6.68.1 Detailed Description

Returns the transpose

6.68.2 Member Function/Subroutine Documentation

6.68.2.1 mqc_matrix_transpose()

```
type(mqc_matrix) function mqc_algebra::transpose::mqc_matrix_transpose (
            class(mqc_matrix), intent(in) Matrix )
```

MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix

Purpose:

```
MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix.
```

Parameters

in	*Matrix*	Matrix is Type(MQC_Matrix) The MQC matrix to be transposed.

Author

L. M. Thompson

X. Sheng

Date

2016, 2017

6.68.2.2 mqc_vector_transpose()

```
type(mqc_vector) function mqc_algebra::transpose::mqc_vector_transpose (
            class(mqc_vector), intent(in) Vector )
```

MQC_Vector_Transpose is a function that returns the transpose of an MQC vector

Purpose:

```
MQC_Vector_Transpose is a function that returns the transpose of an MQC vector.
```

Parameters

in	*Vector*	Vector is Type(MQC_Vector) The MQC vector to transpose.

Author

H. P. Hratchian

Date

2016

The documentation for this interface was generated from the following file:

- src/mqc_algebra.F03

Chapter 7

File Documentation

7.1 src/mqc_algebra.F03 File Reference

Data Types

- type mqc_algebra::mqc_scalar
    Rank 0 array variable
- type mqc_algebra::mqc_vector
    Rank 1 array variable
- type mqc_algebra::mqc_matrix
    Rank 2 array variable
- type mqc_algebra::mqc_r4tensor
    Updates the specified element of the MQC Matrix to the specified value
- interface mqc_algebra::mqc_print
    Prints an object
- interface mqc_algebra::contraction
    Contracts two arrays
- interface mqc_algebra::conjg
    Returns the complex conjugate
- interface mqc_algebra::mqc_have_real
    Determines in an array is real type
- interface mqc_algebra::mqc_have_int
    Determines in an array is integer type
- interface mqc_algebra::mqc_have_complex
    Determines in an array is complex type
- interface mqc_algebra::mqc_cast_integer
    Sets an array to integer type
- interface mqc_algebra::mqc_cast_real
    Sets an array to real type
- interface mqc_algebra::mqc_cast_complex
    Sets an array to complex type
- interface mqc_algebra::matmul

Multiplies two arrays
- interface mqc_algebra::transpose

  Returns the transpose
- interface mqc_algebra::dagger

  Returns the Hermitian conjugate
- interface mqc_algebra::cmplx

  Defines a complex number
- interface mqc_algebra::sqrt

  Returns the square root
- interface mqc_algebra::abs

  Takes the absolute value
- interface mqc_algebra::real

  Returns the real part
- interface mqc_algebra::aimag

  Returns the imaginary part
- interface mqc_algebra::sin

  Returns the sine
- interface mqc_algebra::cos

  Returns the cosine
- interface mqc_algebra::tan

  Returns the tangent
- interface mqc_algebra::asin

  Returns the arcsine
- interface mqc_algebra::acos

  Returns the arccosine
- interface mqc_algebra::atan

  Returns the arctangent
- interface mqc_algebra::atan2

  Returns the arctangent accounting for circle quadrant
- interface mqc_algebra::mqc_set_array2vector

  Sets an intrinsic array as an MQC Algebra object
- interface mqc_algebra::mqc_matrix_symmmatrix_put

  Sets a symmetric packed intrinsic array as an MQC Matrix object
- interface mqc_algebra::mqc_matrix_diagmatrix_put

  Sets a diagonal packed intinsic array as an MQC Matrix object
- interface mqc_algebra::matrix_symm2sq

  Sets a symmetric packed intrinsic array as a square packed intrinsic array
- interface mqc_algebra::dot_product

  Returns the dot product
- interface mqc_algebra::assignment(=)

  Assigns a variable to the value of another
- interface mqc_algebra::operator(+)

  Sums two variables
- interface mqc_algebra::operator(-)

  Subtracts two variables
- interface mqc_algebra::operator(∗)

  Multiplies two variables

- interface mqc_algebra::operator(/)

    Divides two variables
- interface mqc_algebra::operator(∗∗)

    Exponentials a variable to the power of another
- interface mqc_algebra::operator(.ne.)

    Determines if two variables are not equal
- interface mqc_algebra::operator(.eq.)

    Determines if two variables are equal
- interface mqc_algebra::operator(.lt.)

    Determines if a variable is less than another
- interface mqc_algebra::operator(.gt.)

    Determines if a variable is greater than another
- interface mqc_algebra::operator(.le.)

    Determines if a variable is less than or equal to another
- interface mqc_algebra::operator(.ge.)

    Determines if a variable is greater than or equal to another
- interface mqc_algebra::assignment(=)

    Assigns a variable to the value of another
- interface mqc_algebra::operator(.dot.)

    Computes the inner product of two arrays
- interface mqc_algebra::operator(∗)

    Multiplies two variables
- interface mqc_algebra::operator(/)

    Divides two variables
- interface mqc_algebra::operator(+)

    Sums two variables
- interface mqc_algebra::operator(-)

    Subtracts two variables
- interface mqc_algebra::operator(.ewp.)

    Computes the element-wise product of two arrays
- interface mqc_algebra::operator(.ewd.)

    Computes the element-wise quotient of two arrays
- interface mqc_algebra::operator(.x.)

    Computes the cross product of two vectors
- interface mqc_algebra::operator(.outer.)

    Computes the outer product of two vectors
- interface mqc_algebra::assignment(=)

    Assigns a variable to the value of another
- interface mqc_algebra::operator(+)

    Sums two variables
- interface mqc_algebra::operator(-)

    Subtracts two variables
- interface mqc_algebra::operator(∗)

    Multiplies two variables
- interface mqc_algebra::operator(.dot.)

    Computes the inner product of two arrays
- interface mqc_algebra::assignment(=)

    Assigns a variable to the value of another

Modules

- module mqc_algebra

    MQC Algebra contains mathematical objects that are designed to simplify and automate variable use in Fortran

Functions/Subroutines

- integer(kind=int64) function mqc_algebra::factorial (n)

    Factorial returns the factorial of an integer
- integer(kind=int64) function mqc_algebra::bin_coeff (N, K)

    Bin_Coeff returns the binomial coefficient of (n,k)
- subroutine mqc_algebra::mqc_allocate_scalar (Scalar, Data_type)

    MQC_Allocate_Scalar is used to allocate a scalar type variable of the MQC_Scalar class
- subroutine mqc_algebra::mqc_deallocate_scalar (Scalar)

    MQC_Deallocate_Scalar is used to deallocate a scalar type variable of the MQC_Scalar class
- logical function mqc_algebra::mqc_scalar_isallocated (Scalar)

    MQC_Scalar_IsAllocated is used to determine the allocation status of an MQC_Scalar
- subroutine mqc_algebra::mqc_input_integer_scalar (ScalarOut, ScalarIn)

    MQC_Input_Integer_Scalar is a subroutine is used to set an intrinsic integer to an MQC_Scalar
- subroutine mqc_algebra::mqc_input_real_scalar (ScalarOut, ScalarIn)

    MQC_Input_Real_Scalar is a subroutine is used to set an intrinsic real to an MQC_Scalar
- subroutine mqc_algebra::mqc_input_complex_scalar (ScalarOut, ScalarIn)

    MQC_Input_Complex_Scalar is a subroutine is used to set an intrinsic complex to an MQC_Scalar
- subroutine mqc_algebra::mqc_output_mqcscalar_scalar (ScalarOut, ScalarIn)

    MQC_Output MQCScalar_Scalar is a subroutine used to output an MQC_scalar equal to an MQC_Scalar
- subroutine mqc_algebra::mqc_output_integer_scalar (ScalarOut, ScalarIn)

    MQC_Output_Integer_Scalar is a subroutine used to output an intrinsic integer equal to an MQC_Scalar
- subroutine mqc_algebra::mqc_output_real_scalar (ScalarOut, ScalarIn)

    MQC_Output_Real_Scalar is a subroutine used to output an intrinsic real equal to an MQC_Scalar
- subroutine mqc_algebra::mqc_output_complex_scalar (ScalarOut, ScalarIn)

    MQC_Output_Complex_Scalar is a subroutine used to output an intrinsic complex equal to an MQC_Scalar
- subroutine mqc_algebra::mqc_print_scalar_algebra1 (Scalar, IOut, Header, Blank_At_Top, Blank_At_Bottom)

    MQC_Print_Scalar_Algebra1 is a subroutine used to print an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_cmplx (Scalar1, Scalar2)

    MQC_Scalar_Cmplx is a function used to set a complex MQC_Scalar type variable from two other MQC_scalars
- type(mqc_scalar) function mqc_algebra::mqc_scalar_sqrt (Scalar)

    MQC_Scalar_Sqrt is a function used to return the square root of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_sin (Scalar)

    MQC_Scalar_Sin is a function used to return the sine of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_cos (Scalar)

    MQC_Scalar_Cos is a function used to return the cosine of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_tan (Scalar)

    MQC_Scalar_Tan is a function used to return the tangent of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_asin (Scalar)

    MQC_Scalar_ASin is a function used to return the arcsin of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_acos (Scalar)

    MQC_Scalar_ACos is a function used to return the arccosine of an MQC_scalar

- type(mqc_scalar) function mqc_algebra::mqc_scalar_atan (Scalar)

  MQC_Scalar_ATan is a function used to return the arctangent of an MQC_scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_atan2 (Scalar)

  MQC_Scalar_ATan2 is a function used to return the arctangent of an MQC_scalar accounting for quadrant of Argand diagram
- logical function mqc_algebra::mqc_scalar_havereal (Scalar)

  MQC_Scalar_HaveReal is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type real
- logical function mqc_algebra::mqc_scalar_haveinteger (Scalar)

  MQC_Scalar_HaveInteger is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type integer
- logical function mqc_algebra::mqc_scalar_havecomplex (Scalar)

  MQC_Scalar_HaveComplex is a function that returns TRUE or FALSE indicating whether an MQC_scalar is of type complex
- real(kind=real64) function mqc_algebra::mqc_scalar_get_intrinsic_real (Scalar)

  MQC_Scalar_Get_Intrinsic_Real is a function that returns the MQC_scalar value as an intrinsic real
- integer(kind=int64) function mqc_algebra::mqc_scalar_get_intrinsic_integer (Scalar)

  MQC_Scalar_Get_Intrinsic_Integer is a function that returns the MQC_scalar value as an intrinsic integer
- complex(kind=real64) function mqc_algebra::mqc_scalar_get_intrinsic_complex (Scalar)

  MQC_Scalar_Get_Intrinsic_Complex is a function that returns the MQC_scalar value as an intrinsic complex
- type(mqc_scalar) function mqc_algebra::mqc_scalar_get_abs_value (Scalar)

  MQC_Scalar_Get_ABS_Value is a function that returns the absolute value of MQC_scalar variable
- subroutine mqc_algebra::mqc_scalar_get_random_value (Scalar, Seed, Distribution)

  MQC_Scalar_Get_Random_Value is a function that returns a random real value from a specified distribution
- type(mqc_scalar) function mqc_algebra::mqc_scalaradd (Scalar1, Scalar2)

  MQC_ScalarAdd is a function that sums two MQC_Scalar objects
- type(mqc_scalar) function mqc_algebra::mqc_scalarsubtract (Scalar1, Scalar2)

  MQC_ScalarSubtract is a function that subtracts two MQC_Scalar objects
- type(mqc_scalar) function mqc_algebra::mqc_scalarmultiply (Scalar1, Scalar2)

  MQC_ScalarMultiply is a function that multiplies two MQC_Scalar objects
- type(mqc_scalar) function mqc_algebra::mqc_scalardivide (Scalar1, Scalar2)

  MQC_ScalarDivide is a function that divides two MQC_Scalar objects
- type(mqc_scalar) function mqc_algebra::mqc_scalarexponent (Scalar1, Scalar2)

  MQC_ScalarExponent is a function that raises one MQC_Scalar to the power of another MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarintegerexponent (Scalar, IntIn)

  MQC_ScalarIntegerExponent is a function that raises an MQC_Scalar to the power of an intrinsic integer
- type(mqc_scalar) function mqc_algebra::mqc_scalarrealexponent (Scalar, 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, CompIn)

  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)

    MQC_RealGTScalar is a function that returns TRUE if an intrinsic real is greater than a MQC_Scalar
- logical function mqc_algebra::mqc_scalargtreal (Scalar, RealIn)

    MQC_ScalarGTReal is a function that returns TRUE if a MQC_Scalar is greater than an intrinsic real
- logical function mqc_algebra::mqc_scalarle (Scalar1, Scalar2)

    MQC_ScalarLE is a function that returns TRUE if the left MQC_Scalar is less than or equal the right MQC_Scalar
- logical function mqc_algebra::mqc_reallescalar (RealIn, Scalar)

    MQC_RealLEScalar is a function that returns TRUE if an intrinsic real is less than or equal to a MQC_Scalar
- logical function mqc_algebra::mqc_scalarlereal (Scalar, RealIn)

    MQC_ScalarLEReal is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic real
- logical function mqc_algebra::mqc_integerlescalar (IntIn, Scalar)

    MQC_IntegerLEScalar is a function that returns TRUE if an intrinsic integer is less than or equal to a MQC_Scalar
- logical function mqc_algebra::mqc_scalarleinteger (Scalar, IntIn)

    MQC_ScalarLEInteger is a function that returns TRUE if a MQC_Scalar is less than or equal to an intrinsic integer
- logical function mqc_algebra::mqc_scalarge (Scalar1, Scalar2)

    MQC_ScalarGE is a function that returns TRUE if the left MQC_Scalar is greater than or equal the right MQC_↩ Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_conjugate (ScalarIn)

    MQC_Scalar_Complex_Conjugate is a function that returns the complex conjugate of an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_realpart (ScalarIn)

    MQC_Scalar_Complex_RealPart is a function that returns the real part of an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalar_complex_imagpart (ScalarIn)

    MQC_Scalar_Complex_ImagPart is a function that returns the inaginary part of an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_integerscalarmultiply (IntegerIn, Scalar)

    MQC_IntegerScalarMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarintegermultiply (Scalar, IntegerIn)

    MQC_ScalarIntegerMultiply is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_realscalarmultiply (RealIn, Scalar)

    MQC_RealScalarMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarrealmultiply (Scalar, RealIn)

    MQC_ScalarRealMultiply is a function that is used to multiply an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_complexscalarmultiply (ComplexIn, Scalar)

    MQC_ComplexScalarMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexmultiply (Scalar, ComplexIn)

    MQC_ScalarComplexMultiply is a function that is used to multiply an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_integerscalardivide (IntegerIn, Scalar)

    MQC_IntegerScalarDivide is a function that is used to divide an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarintegerdivide (Scalar, IntegerIn)

    MQC_ScalarIntegerDivide is a function that is used to divide an MQC_Scalar by an intrinsic integer

- type(mqc_scalar) function mqc_algebra::mqc_realscalardivide (RealIn, Scalar)

    MQC_RealScalarDivide is a function that is used to divide an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarrealdivide (Scalar, RealIn)

    MQC_ScalarRealDivide is a function that is used to divide an MQC_Scalar by an intrinsic real
- type(mqc_scalar) function mqc_algebra::mqc_complexscalardivide (ComplexIn, Scalar)

    MQC_ComplexScalarDivide is a function that is used to divide an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexdivide (Scalar, ComplexIn)

    MQC_ScalarComplexDivide is a function that is used to divide an MQC_Scalar by an intrinsic complex
- type(mqc_scalar) function mqc_algebra::mqc_integerscalaradd (IntegerIn, Scalar)

    MQC_IntegerScalarAdd is a function that is used to multiply an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarintegeradd (Scalar, IntegerIn)

    MQC_ScalarIntegerAdd is a function that is used to sum an intrinsic integer by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_realscalaradd (RealIn, Scalar)

    MQC_RealScalarAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarrealadd (Scalar, RealIn)

    MQC_ScalarRealAdd is a function that is used to sum an intrinsic real by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_complexscalaradd (ComplexIn, Scalar)

    MQC_ComplexScalarAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexadd (Scalar, ComplexIn)

    MQC_ScalarComplexAdd is a function that is used to sum an intrinsic complex by an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_integerscalarsubtract (IntegerIn, Scalar)

    MQC_IntegerScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic integer
- type(mqc_scalar) function mqc_algebra::mqc_scalarintegersubtract (Scalar, IntegerIn)

    MQC_ScalarIntegerSubtract is a function that is used to subtract an intrinsic integer from an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_realscalarsubtract (RealIn, Scalar)

    MQC_RealScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic real
- type(mqc_scalar) function mqc_algebra::mqc_scalarrealsubtract (Scalar, RealIn)

    MQC_ScalarRealSubtract is a function that is used to subtract an intrinsic real from an MQC_Scalar
- type(mqc_scalar) function mqc_algebra::mqc_complexscalarsubtract (ComplexIn, Scalar)

    MQC_ComplexScalarSubtract is a function that is used to subtract an MQC_Scalar from an intrinisic complex
- type(mqc_scalar) function mqc_algebra::mqc_scalarcomplexsubtract (Scalar, ComplexIn)

    MQC_ScalarComplexSubtract is a function that is used to subtract an intrinsic complex from an MQC_Scalar
- subroutine mqc_algebra::mqc_allocate_vector (N, Vector, Data_Type)

    MQC_Allocate_Vector is used to allocate a vector type variable of the MQC_Vector class
- subroutine mqc_algebra::mqc_deallocate_vector (Vector)

    MQC_Deallocate_Vector is used to deallocate a vector type variable of the MQC_Vector class
- integer(kind=int64) function mqc_algebra::mqc_length_vector (Vector)

    MQC_Length_Vector is used to return the length of an MQC vector
- logical function mqc_algebra::mqc_vector_havereal (Vector)

    MQC_Vector_HaveReal is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated real vector
- logical function mqc_algebra::mqc_vector_haveinteger (Vector)

    MQC_Vector_HaveInteger is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated integer vector
- logical function mqc_algebra::mqc_vector_havecomplex (Vector)

    MQC_Vector_HaveComplex is a function that returns TRUE or FALSE indicating whether the MQC vector has an allocated complex vector
- logical function mqc_algebra::mqc_vector_iscolumn (Vector)

MQC_Vector_IsColumn is a function that returns TRUE if the MQC vector is a column vector and FALSE if the MQC vector is a row vector

- subroutine mqc_algebra::mqc_vector_copy_int2real (Vector)

  MQC_Vector_Copy_Int2Real is a subroutine that copies an integer MQC_Vector into its real vector
- subroutine mqc_algebra::mqc_vector_copy_int2complex (Vector)

  MQC_Vector_Copy_Int2Complex is a subroutine that copies an integer MQC_Vector into its complex vector
- subroutine mqc_algebra::mqc_vector_copy_real2int (Vector)

  MQC_Vector_Copy_Real2Int is a subroutine that copies a real MQC_Vector into its integer vector
- subroutine mqc_algebra::mqc_vector_copy_real2complex (Vector)

  MQC_Vector_Copy_Real2Complex is a subroutine that copies a real MQC_Vector into its complex vector
- subroutine mqc_algebra::mqc_vector_copy_complex2int (Vector)

  MQC_Vector_Copy_Complex2Int is a subroutine that copies a complex MQC_Vector into its integer vector
- subroutine mqc_algebra::mqc_vector_copy_complex2real (Vector)

  MQC_Vector_Copy_Complex2Real is a subroutine that copies a complex MQC_Vector into its real vector
- type(mqc_scalar) function mqc_algebra::mqc_vector_scalar_at (Vec, I)

  MQC_Vector_Scalar_At is a function that returns the ith element of a MQC vector as an MQC scalar
- type(mqc_vector) function mqc_algebra::mqc_vector_vector_at (Vec, I, J)

  MQC_Vector_Vector_At is a function that returns the vector at the specified subvector of MQC_Vector
- subroutine mqc_algebra::mqc_set_vector2integerarray (ArrayOut, VectorIn)

  MQC_Set_Vector2IntegerArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic integer array
- subroutine mqc_algebra::mqc_set_vector2realarray (ArrayOut, VectorIn)

  MQC_Set_Vector2RealArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic real array
- subroutine mqc_algebra::mqc_set_vector2complexarray (ArrayOut, VectorIn)

  MQC_Set_Vector2ComplexArray is a subroutine that outputs an MQC vector to a rank 1 intrinsic complex array
- subroutine mqc_algebra::mqc_set_array2vector_integer (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Integer is a subroutine that sets a rank 1 intrinsic integer array equal to a MQC vector
- subroutine mqc_algebra::mqc_set_array2vector_real (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Real is a subroutine that sets a rank 1 vector intrinsic real array equal to a MQC vector
- subroutine mqc_algebra::mqc_set_array2vector_complex (VectorOut, ArrayIn)

  MQC_Set_Array2Vector_Complex is a subroutine that sets a rank 1 vector intrinsic complex array equal to a MQC vector
- subroutine mqc_algebra::mqc_set_vector2vector (VectorOut, VectorIn)

  MQC_Set_Vector2Vector is a subroutine that sets a MQC vector equal to another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vectorvectorsum (Vector1In, Vector2In)

  MQC_VectorVectorSum is a function that adds two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vectorvectordifference (Vector1In, Vector2In)

  MQC_VectorVectorDifference is a function that subtracts two MQC vectors and stores them in another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_scalarvectorsum (ScalarIn, VectorIn)

  MQC_ScalarVectorSum is a function that adds an MQC scalar to all elements of an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_scalarvectordifference (ScalarIn, VectorIn)

  MQC_ScalarVectorDifference is a function that subtracts an MQC scalar from all elements of an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_elementvectorproduct (Vector1In, Vector2In)

  MQC_ElementVectorProduct is a function that multiplies two MQC vectors elementwise and stores them into another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_transpose (Vector)

  MQC_Vector_Transpose is a function that returns the transpose of an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_conjugate_transpose (Vector)

  MQC_Vector_Conjugate_Transpose is a function that returns the conjugate transpose of an MQC vector

- type(mqc_scalar) function mqc_algebra::mqc_vectorvectordotproduct (Vector1, Vector2)

  MQC_VectorVectorDotProduct is a function that returns the dot product of two MQC vectors
- type(mqc_matrix) function mqc_algebra::mqc_outer (VA, VB)

  MQC_Outer is a function that returns the outer product of two MQC vectors
- type(mqc_vector) function mqc_algebra::mqc_crossproduct (Vector1In, Vector2In)

  MQC_CrossProduct is a function that returns the cross product of two MQC vectors
- subroutine mqc_algebra::mqc_print_vector_algebra1 (Vector, IOut, Header, Verbose, Blank_At_Top, Blank_At←_Bottom)

  MQC_Print_Vector_Algebra1 is a subroutine used to print an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_cast_integer (VA)

  MQC_vector_cast_integer is a function that converts an MQC vector to its integer space
- type(mqc_vector) function mqc_algebra::mqc_vector_cast_real (VA)

  MQC_vector_cast_real is a function that converts an MQC vector to its real space
- type(mqc_vector) function mqc_algebra::mqc_vector_cast_complex (VA)

  MQC_vector_cast_complex is a function that converts an MQC vector to its complex space
- subroutine mqc_algebra::mqc_vector_scalar_put (Vector, Scalar, I)

  MQC_Vector_Scalar_Put is a subroutine that updates the value of the ith element of a MQC vector with the value of a MQC scalar
- subroutine mqc_algebra::mqc_vector_scalar_increment (Vector, Scalar, I)

  MQC_Vector_Scalar_Increment is a subroutine that increments the value of the ith element of a MQC vector by the value of a MQC scalar
- subroutine mqc_algebra::mqc_vector_vector_put (Vector, VectorIn, I)

  MQC_Vector_Vector_Put is a subroutine that updates the values of a subvector of a MQC vector with the values of a MQC vector
- subroutine mqc_algebra::mqc_vector_initialize (Vector, Length, Scalar)

  MQC_Vector_Initialize is a subroutine that initializes a MQC vector
- type(mqc_vector) function mqc_algebra::mqc_scalarvectorproduct (Scalar, Vector)

  MQC_ScalarVectorProduct is a function that returns the product of a MQC scalar with a MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vectorscalarproduct (vector, scalar)

  MQC_VectorScalarProduct is a function that returns the product of a MQC vector with a MQC scalar
- type(mqc_vector) function mqc_algebra::mqc_vectorscalardivide (vector, scalar)

  MQC_VectorScalarDivide is a function that returns a MQC vector divided by a MQC scalar
- type(mqc_vector) function mqc_algebra::mqc_realvectorproduct (RealIn, Vector)

  MQC_RealVectorProduct is a function that returns the product of an intrinsic real scalar and a MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vectorrealproduct (vector, realIn)

  MQC_VectorRealProduct is a function that returns the product of a MQC vector and an intrinsic real scalar
- type(mqc_vector) function mqc_algebra::mqc_vectorrealdivide (vector, realIn)

  MQC_VectorRealDivide is a function that returns a MQC vector divided by an intrinsic real integer
- type(mqc_vector) function mqc_algebra::mqc_integervectorproduct (intIn, Vector)

  MQC_IntegerVectorProduct is a function that returns the product of an intrinsic integer scalar and a MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vectorintegerproduct (vector, intIn)

  MQC_VectorIntegerProduct is a function that returns the product of a MQC vector and an intrinsic integer scalar
- type(mqc_vector) function mqc_algebra::mqc_vectorintegerdivide (vector, intIn)

  MQC_VectorIntegerDivide is a function that returns a MQC vector divided by an intrinsic integer scalar
- type(mqc_vector) function mqc_algebra::mqc_complexvectorproduct (CompIn, Vector)

MQC_ComplexVectorProduct is a function that returns the product of an intrinsic complex scalar and a MQC vector

- type(mqc_vector) function mqc_algebra::mqc_vectorcomplexproduct (vector, compIn)

  MQC_VectorComplexProduct is a function that returns the product of a MQC vector and an intrinsic complex scalar
- type(mqc_vector) function mqc_algebra::mqc_vectorcomplexdivide (vector, compIn)

  MQC_VectorComplexDivide is a function that returns a MQC vector divided by an intrinsic complex scalar
- type(mqc_scalar) function mqc_algebra::mqc_vector_norm (vector, methodIn)

  MQC_Vector_Norm is a function that returns the norm of an MQC vector
- logical function mqc_algebra::mqc_vector_isallocated (Vector)

  MQC_Vector_isAllocated is a function that returns TRUE is an MQC vector is allocated and FALSE if it is not
- subroutine mqc_algebra::mqc_vector_push (Vector, Scalar)

  MQC_Vector_Push is a function that adds a value to the end of a MQC vector
- subroutine mqc_algebra::mqc_vector_unshift (Vector, Scalar)

  MQC_Vector_Unshift is a function that adds a value to the beginning of a MQC vector
- type(mqc_scalar) function mqc_algebra::mqc_vector_pop (Vector)

  MQC_Vector_Pop is a function that removes a value from the end of a MQC vector and returns it
- type(mqc_scalar) function mqc_algebra::mqc_vector_shift (Vector)

  MQC_Vector_Shift is a function that removes a value from the beginning of a MQC vector and returns it
- type(mqc_scalar) function mqc_algebra::mqc_vector_maxval (Vector)

  MQC_Vector_MaxVal is a function that returns the largest value in an MQC vector
- type(mqc_scalar) function mqc_algebra::mqc_vector_minval (Vector)

  MQC_Vector_MinVal is a function that returns the smallest value in an MQC vector
- integer function mqc_algebra::mqc_vector_maxloc (Vector)

  MQC_Vector_MaxLoc is a function that returns the index of the largest value in an MQC vector
- integer function mqc_algebra::mqc_vector_minloc (Vector)

  MQC_Vector_MinLoc is a function that returns the index of the smallest value in an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_argsort (Vector)

  MQC_Vector_Argsort is a function that returns the indices of an an MQC vector sorted from low to high
- subroutine mqc_algebra::mqc_vector_sort (Vector, idx)

  MQC_Vector_Sort is a function that returns an MQC vector sorted from low to high unless optional index order is present
- subroutine mqc_algebra::mqc_vector_sqrt (A)

  MQC_Vector_Sqrt is a function that returns the square root of all elements of an MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_abs (A)

  MQC_Vector_Abs is a function that returns the absolute value of all elements of an MQC vector
- subroutine mqc_algebra::mqc_vector_power (A, P)

  MQC_Vector_Power is a function that returns the value of all elements of an MQC vector raised to a power
- type(mqc_vector) function mqc_algebra::mqc_vector_complex_realpart (A)

  MQC_Vector_Complex_RealPart is a function that returns a MQC vector with elements containing the real part of elements of another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_complex_imagpart (A)

  MQC_Vector_Complex_ImagPart is a function that returns a MQC vector with elements containing the imaginary part of elements of another MQC vector
- type(mqc_vector) function mqc_algebra::mqc_vector_cmplx (Vector1, Vector2)

  MQC_Vector_Cmplx is a function that takes a MQC vector representing the real part and a MQC vector representing the imaginary part and combines them into another MQC vector

- subroutine mqc_algebra::mqc_matrix_diagonalize (A, EVals, EVecs)

    MQC_Matrix_Diagonalize is a subroutine that takes a symmetric or hermitian MQC matrix and returns eigenvalues and eigenvectors
- type(mqc_matrix) function mqc_algebra::mqc_matrix_cast_real (MA)

    MQC_Matrix_Cast_Real is a function that converts an MQC matrix to its real space
- type(mqc_matrix) function mqc_algebra::mqc_matrix_cast_integer (MA)

    MQC_Matrix_Cast_Integer is a function that converts an MQC matrix to its integer space
- type(mqc_matrix) function mqc_algebra::mqc_matrix_cast_complex (MA)

    MQC_Matrix_Cast_Complex is a function that converts an MQC matrix to its complex space
- type(mqc_scalar) function mqc_algebra::mqc_matrix_scalar_at (Mat, I, J)

    MQC_Matrix_Scalar_At is a function that returns the value of an element of a MQC matrix
- type(mqc_vector) function mqc_algebra::mqc_matrix_vector_at (Mat, Rows, Cols)

    MQC_Matrix_Vector_At is a function that returns the subvector of an MQC matrix
- recursive subroutine mqc_algebra::mqc_matrix_vector_put (Mat, VectorIn, Rows, Cols)

    MQC_Matrix_Vector_Put is a subroutine that writes a subvector to the specified position of a MQC matrix
- type(mqc_matrix) function mqc_algebra::mqc_matrix_matrix_at (Mat, Rows, Cols)

    MQC_Matrix_Matrix_At is a function that returns a submatrix of the matrix
- subroutine mqc_algebra::mqc_matrix_diagmatrix_put_vector (diagVectorIn, mat)

    MQC_Matrix_DiagMatrix_Put_Vector is a subroutine that returns a diagonal MQC matrix with elements defined by values in a MQC vector
- subroutine mqc_algebra::mqc_matrix_diagmatrix_put_integer (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_integer is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_algebra::mqc_matrix_diagmatrix_put_real (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_Real is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_algebra::mqc_matrix_diagmatrix_put_complex (mat, diagMatrixIn)

    MQC_Matrix_DiagMatrix_Put_Complex is a subroutine that returns a diagonal MQC matrix with elements defined by values in an intrinsic complex vector
- subroutine mqc_algebra::mqc_matrix_symmmatrix_put_integer (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Integer is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic integer vector
- subroutine mqc_algebra::mqc_matrix_symmmatrix_put_real (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Real is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic real vector
- subroutine mqc_algebra::mqc_matrix_symmmatrix_put_complex (mat, symmMatrixIn)

    MQC_Matrix_SymmMatrix_Put_Complex is a subroutine that returns a symmetric packed MQC matrix with elements defined by values in an intrinsic complex vector
- recursive subroutine mqc_algebra::mqc_matrix_matrix_put (Mat, MatrixIn, Rows, Cols)

    MQC_Matrix_Matrix_Put is a subroutine that writes a submatrix to the specified position of a MQC matrix
- integer(kind=int64) function mqc_algebra::symindexhash (i, j, k, l)

    SymIndexHash is a function that returns the index in a vector of a symmetric-packed matrix or rank-4 tensor
- type(mqc_matrix) function mqc_algebra::mqc_elementmatrixproduct (A, B)

    MQC_ElementMatrixProduct is a function that returns the element- wise product of two MQC matrices
- type(mqc_matrix) function mqc_algebra::mqc_elementmatrixdivide (A, B)

    MQC_ElementMatrixDivide is a function that returns the element- wise quotient of two MQC matrices
- logical function mqc_algebra::mqc_matrix_test_symmetric (Matrix, Option)

MQC_Matrix_Test_Symmetric is a function that tests a MQC matrix for symmetry
- logical function mqc_algebra::mqc_matrix_test_diagonal (Matrix)

  MQC_Matrix_Test_Diagonal is a function that tests a MQC matrix to determine if it is diagonal
- subroutine mqc_algebra::mqc_allocate_matrix (M, N, Matrix, Data_Type, Storage)

  MQC_Allocate_Matrix is used to allocate a matrix type variable of the MQC_Matrix class
- subroutine mqc_algebra::mqc_deallocate_matrix (Matrix)

  MQC_Deallocate_Matrix is used to deallocate a matrix type variable of the MQC_Matrix class
- logical function mqc_algebra::mqc_matrix_isallocated (Matrix)

  MQC_Matrix_isAllocate is a function that returns the allocation status of a MQC_Matrix variable
- subroutine mqc_algebra::mqc_set_integerarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_IntegerArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic integer rank-2 array
- subroutine mqc_algebra::mqc_set_realarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_RealArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic real rank-2 array
- subroutine mqc_algebra::mqc_set_complexarray2matrix (MatrixOut, ArrayIn)

  MQC_Set_ComplexArray2Matrix is a subroutine that sets an MQC matrix equal to an intrinsic complex rank-2 array
- subroutine mqc_algebra::mqc_set_matrix2integerarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2IntegerArray is a subroutine that sets an intrinsic integer rank-2 array equal to an MQC matrix
- subroutine mqc_algebra::mqc_set_matrix2realarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2RealArray is a subroutine that sets an intrinsic real rank-2 array equal to an MQC matrix
- subroutine mqc_algebra::mqc_set_matrix2complexarray (ArrayOut, MatrixIn)

  MQC_Set_Matrix2ComplexArray is a subroutine that sets an intrinsic complex rank-2 array equal to an MQC matrix
- subroutine mqc_algebra::mqc_set_matrix2matrix (MatrixOut, MatrixIn)

  MQC_Set_Matrix2Matrix is a subroutine that sets an MQC matrix equal to another MQC matrix
- subroutine mqc_algebra::mqc_print_matrix_algebra1 (Matrix, IOut, Header, Blank_At_Top, Blank_At_Bottom)

  MQC_Print_Matrix_Algebra1 is a subroutine used to print an MQC matrix
- subroutine mqc_algebra::mqc_matrix_copy_int2real (Matrix)

  MQC_Matrix_Copy_Int2Real is a subroutine used to copy an integer MQC matrix into its real space
- subroutine mqc_algebra::mqc_matrix_copy_int2complex (Matrix)

  MQC_Matrix_Copy_Int2Complex is a subroutine used to copy an integer MQC matrix into its complex space
- subroutine mqc_algebra::mqc_matrix_copy_real2int (Matrix)

  MQC_Matrix_Copy_Real2Int is a subroutine used to copy a real MQC matrix into its integer space
- subroutine mqc_algebra::mqc_matrix_copy_real2complex (Matrix)

  MQC_Matrix_Copy_Real2Complex is a subroutine used to copy a real MQC matrix into its complex space
- subroutine mqc_algebra::mqc_matrix_copy_complex2int (Matrix)

  MQC_Matrix_Copy_Complex2Int is a subroutine used to copy a complex MQC matrix into its integer space
- subroutine mqc_algebra::mqc_matrix_copy_complex2real (Matrix)

  MQC_Matrix_Copy_Complex2Real is a subroutine used to copy a complex MQC matrix into its real space
- integer(kind=int64) function mqc_algebra::mqc_matrix_rows (Matrix)

  MQC_Matrix_Rows is a function used to return the number of rows of an MQC matrix
- integer(kind=int64) function mqc_algebra::mqc_matrix_columns (Matrix)

  MQC_Matrix_Columns is a function used to return the number of columns of an MQC matrix
- logical function mqc_algebra::mqc_matrix_havereal (Matrix)

  MQC_Matrix_HaveReal is a function used to indicate if an MQC matrix has an allocated real matrix
- logical function mqc_algebra::mqc_matrix_haveinteger (Matrix)

  MQC_Matrix_HaveInteger is a function used to indicate if an MQC matrix has an allocated integer matrix

- logical function mqc_algebra::mqc_matrix_havecomplex (Matrix)

    MQC_Matrix_HaveComplex is a function used to indicate if an MQC matrix has an allocated complex matrix
- logical function mqc_algebra::mqc_matrix_havefull (Matrix)

    MQC_Matrix_HaveFull is a function used to indicate if an MQC matrix is stored unpacked
- logical function mqc_algebra::mqc_matrix_havesymmetric (Matrix)

    MQC_Matrix_HaveSymmetric is a function used to indicate if an MQC matrix is stored symmetric-packed
- logical function mqc_algebra::mqc_matrix_havediagonal (Matrix)

    MQC_Matrix_HaveDiagonal is a function used to indicate if an MQC matrix is stored diagonal-packed
- type(mqc_matrix) function mqc_algebra::mqc_matrix_transpose (Matrix)

    MQC_Matrix_Transpose is a function that returns the transpose of a MQC matrix
- type(mqc_matrix) function mqc_algebra::mqc_matrix_conjugate_transpose (Matrix)

    MQC_Matrix_Conjugate_Transpose is a function that returns the conjugate transpose of a MQC matrix
- type(mqc_matrix) function mqc_algebra::mqc_matrix_symmetrize (Matrix)

    MQC_Matrix_Symmetrize is a function that symmetrizes a MQC matrix
- subroutine mqc_algebra::mqc_matrix_full2symm (Matrix)

    MQC_Matrix_Full2Symm is a subroutine that converts an unpacked MQC matrix to symmetric-packed
- subroutine mqc_algebra::mqc_matrix_symm2full (Matrix, Option)

    MQC_Matrix_Symm2Full is a subroutine that converts a symmetry-packed MQC matrix to unpacked
- subroutine mqc_algebra::mqc_matrix_full2diag (Matrix)

    MQC_Matrix_Full2Diag is a subroutine that converts an unpacked MQC matrix to diagonal-packed
- subroutine mqc_algebra::mqc_matrix_diag2full (Matrix)

    MQC_Matrix_Diag2Full is a subroutine that converts a diagonal-packed MQC matrix to unpacked
- subroutine mqc_algebra::mqc_matrix_symm2diag (Matrix)

    MQC_Matrix_Symm2Diag is a subroutine that converts a symmetry-packed MQC matrix to diagonal-packed
- subroutine mqc_algebra::mqc_matrix_diag2symm (Matrix)

    MQC_Matrix_Diag2Symm is a subroutine that converts a diagonal-packed MQC matrix to symmetry-packed
- type(mqc_matrix) function mqc_algebra::mqc_matrix_symm2full_func (Matrix)

    MQC_Matrix_Symm2Full_Func is a function that converts a symmetric- packed MQC matrix to unpacked
- subroutine mqc_algebra::matrix_symm2sq_integer (N, I_Symm, I_Sq)

    Matrix_Symm2Sq_Integer is a subroutine that converts a symmetric- packed intrinsic integer matrix to a rank-2 intrinsic integer array
- subroutine mqc_algebra::matrix_symm2sq_real (N, A_Symm, A_Sq)

    Matrix_Symm2Sq_Real is a subroutine that converts a symmetric- packed intrinsic real matrix to a rank-2 intrinsic real array
- subroutine mqc_algebra::matrix_symm2sq_complex (N, A_Symm, A_Sq)

    Matrix_Symm2Sq_Complex is a subroutine that converts a symmetric- packed intrinsic complex matrix to a rank-2 intrinsic complex array
- type(mqc_matrix) function mqc_algebra::mqc_vector2diagmatrix (vector)

    MQC_Vector2DiagMatrix is a function that outputs a diagonal MQC matrix with elements defined by an MQC vector
- type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixsum (MA, MB)

    MQC_MatrixMatrixSum is a function that sums two MQC matrices
- type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixsubtract (MA, MB)

    MQC_MatrixMatrixSubtract is a function that subtracts two MQC matrices
- type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixproduct (MA, MB)

    MQC_MatrixMatrixProduct is a function that computes the element- wise product of two MQC matrices
- type(mqc_matrix) function mqc_algebra::mqc_matrixmatrixdotproduct (MA, MB)
- type(mqc_vector) function mqc_algebra::mqc_matrixvectordotproduct (MA, VB)

- type(mqc_vector) function mqc_algebra::mqc_vectormatrixdotproduct (VA, MB)
- type(mqc_matrix) function mqc_algebra::mqc_matrixscalarproduct (Matrix, Scalar)
- type(mqc_matrix) function mqc_algebra::mqc_scalarmatrixproduct (Scalar, Matrix)
- type(mqc_scalar) function mqc_algebra::mqc_matrix_matrix_contraction (Matrix1, Matrix2)
- subroutine mqc_algebra::mqc_matrix_scalar_put (Matrix, Scalar, I, J)
- subroutine mqc_algebra::mqc_matrix_initialize (Matrix, Rows, Columns, Scalar, Storage)
- subroutine mqc_algebra::mqc_matrix_identity (matrix, n, m)
- subroutine mqc_algebra::mqc_matrix_set (matrix, scalar, storage)
- type(mqc_scalar) function mqc_algebra::mqc_matrix_norm (matrix, methodIn)
- type(mqc_scalar) function mqc_algebra::mqc_matrix_determinant (a)
- type(mqc_matrix) function mqc_algebra::mqc_matrix_inverse (a)
- type(mqc_scalar) function mqc_algebra::mqc_matrix_trace (matrix)
- subroutine mqc_algebra::mqc_matrix_generalized_eigensystem (a, bIn, eigenvals, reigenvecs, leigenvecs)
- subroutine mqc_algebra::mqc_matrix_svd (A, EVals, EUVecs, EVVecs)
- subroutine mqc_algebra::mqc_matrix_rms_max (A, rms_A, max_A)
- subroutine mqc_algebra::mqc_matrix_sqrt (A, eVals, eVecs)
- type(mqc_matrix) function mqc_algebra::mqc_givens_matrix (m_size, angle, p, q)
- subroutine mqc_algebra::mqc_allocate_r4tensor (I, J, K, L, Tensor, Data_Type, Storage)
- subroutine mqc_algebra::mqc_deallocate_r4tensor (Tensor)
- type(mqc_scalar) function mqc_algebra::mqc_r4tensor_at (Tensor, I, J, K, L)
- subroutine mqc_algebra::mqc_r4tensor_put (Tensor, Element, I, J, K, L)
- subroutine mqc_algebra::mqc_print_r4tensor_algebra1 (Tensor, IOut, Header, blank_at_top, blank_at_bottom)
- subroutine mqc_algebra::mqc_set_array2tensor (TensorOut, ArrayIn)
- subroutine mqc_algebra::mqc_r4tensor_initialize (R4Tensor, I, J, K, L, Scalar)
- subroutine mqc_algebra::mqc_matrix_symmsymmr4tensor_put_real (r4Tensor, symmSymmMatrixIn)
- subroutine mqc_algebra::mqc_matrix_symmsymmr4tensor_put_complex (r4Tensor, symmSymmMatrixIn)
- logical function mqc_algebra::mqc_r4tensor_haveinteger (R4Tensor)
- logical function mqc_algebra::mqc_r4tensor_havereal (R4Tensor)
- logical function mqc_algebra::mqc_r4tensor_havecomplex (R4Tensor)

7.2 src/mqc_est.F03 File Reference

Data Types

- type mqc_est::mqc_scf_integral
- type mqc_est::mqc_scf_eigenvalues
- type mqc_est::mqc_wavefunction
- type mqc_est::mqc_pscf_wavefunction
- type mqc_est::mqc_determinant_string
- type mqc_est::mqc_determinant
- type mqc_est::mqc_twoeris
- interface mqc_est::mqc_print
- interface mqc_est::matmul
- interface mqc_est::dot_product
- interface mqc_est::transpose
- interface mqc_est::dagger
- interface mqc_est::contraction
- interface mqc_est::mqc_matrix_undospinblockghf
- interface mqc_est::assignment(=)
- interface mqc_est::operator(+)
- interface mqc_est::operator(-)
- interface mqc_est::operator(*)

Modules

- module mqc_est

Functions/Subroutines

- subroutine mqc_est::mqc_print_wavefunction (wavefunction, iOut, label)
- subroutine mqc_est::mqc_print_integral (integral, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_est::mqc_print_eigenvalues (eigenvalues, iOut, header, blank_at_top, blank_at_bottom)
- subroutine mqc_est::mqc_print_twoeris (twoERIs, iOut, header, blank_at_top, blank_at_bottom)
- logical function mqc_est::mqc_integral_isallocated (Integral)
- logical function mqc_est::mqc_eigenvalues_isallocated (Eigenvalues)
- logical function mqc_est::mqc_integral_has_alpha (integral)
- logical function mqc_est::mqc_integral_has_beta (integral)
- logical function mqc_est::mqc_integral_has_alphabeta (integral)
- logical function mqc_est::mqc_integral_has_betaalpha (integral)
- logical function mqc_est::mqc_eigenvalues_has_alpha (eigenvalues)
- logical function mqc_est::mqc_eigenvalues_has_beta (eigenvalues)
- character(len=64) function mqc_est::mqc_integral_array_type (integral)
- character(len=64) function mqc_est::mqc_eigenvalues_array_type (eigenvalues)
- character(len=64) function mqc_est::mqc_integral_array_name (integral)
- character(len=64) function mqc_est::mqc_eigenvalues_array_name (eigenvalues)
- subroutine mqc_est::mqc_integral_add_name (integral, arrayName)
- subroutine mqc_est::mqc_eigenvalues_add_name (eigenvalues, arrayName)
- integer(kind=int64) function mqc_est::mqc_integral_dimension (integral, label, axis)
- integer(kind=int64) function mqc_est::mqc_eigenvalues_dimension (eigenvalues, label)
- subroutine mqc_est::mqc_twoeris_allocate (twoERIs, storageType, integralType, alpha, beta, alphaBeta, betaAlpha)
- 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 (IOut, IPrint, NBasisIn, NAlphaIn, NBetaIn, Determinants, NCoreIn)
- type(mqc_scalar) function mqc_est::slater_condon (IOut, IPrint, NBasisIn, Determinants, L_A_String, L_B_String, R_A_String, R_B_String, Core_Hamiltonian, ERIs, UHF)
- subroutine mqc_est::twoeri_trans (IOut, IPrint, MO_Coeff, ERIs, MO_ERIs, UHF)
- subroutine mqc_est::mqc_build_ci_hamiltonian (IOut, IPrint, NBasis, Determinants, MO_Core_Ham, MO_ERIs, UHF, CI_Hamiltonian)
- type(mqc_matrix) function mqc_est::get_one_gamma_matrix (iOut, iPrint, nBasisIn, nState, determinants, ci_↩ amplitudes, nCoreIn, nOrbsIn)

Index