

MODULE FM_QUAD_REAL

! FM_quadreal 1.4 David M. Smith Quadruple Precision Real and Complex Support

! This module extends the definition of basic FM types (FM), (IM), and (ZM) so they can interact
! with quadruple precision real and complex variables.

! Warning: This module is needed only when the user's program explicitly declares quadruple
! precision variables. If quad precision is obtained by using a compiler switch
! to change the default real size for the entire program (such as with gfortran's
! -fdefault-real-8 option), then compiling the basic FM package with the same option
! means this module is not needed.

! Not all compilers might support quadruple precision floating-point, but for those that do,
! variables can be declared via the SELECTED_REAL_KIND function.

! For example, when this module was first written, typical computer hardware supported 32-bit
! floats as single precision and 64-bits as double precision. Some compilers offered 128-bit
! quadruple precision implemented in software. This format used 113 bits for the fraction
! part of a floating-point number, giving about 34 significant digits of precision.

! So SELECTED_REAL_KIND(30) could be used to select this quad format.

! The routines in this interface extend basic functions like TO_FM, TO_IM, TO_ZM so they can be
! used with quad real or complex arguments. New conversion functions TO_QUAD and TO_QUAD_Z
! will take FM, IM, or ZM inputs and convert to quad real or complex.

! Other mixed-mode operations, such as assignment (a = b), logical comparisons, and arithmetic
! are also provided. As with the basic FMZM module, assignments and arithmetic may also involve
! 1 or 2-dimensional arrays.

USE FMZM

INTEGER, PARAMETER :: QUAD_FP = SELECTED_REAL_KIND(30)

REAL (QUAD_FP), PARAMETER :: Q_ZERO = 0, Q_ONE = 1

INTERFACE TO_FM

MODULE PROCEDURE FM_Q

MODULE PROCEDURE FM_ZQ

MODULE PROCEDURE FM_Q1

MODULE PROCEDURE FM_ZQ1

MODULE PROCEDURE FM_Q2

MODULE PROCEDURE FM_ZQ2

END INTERFACE

INTERFACE TO_IM

MODULE PROCEDURE IM_Q

MODULE PROCEDURE IM_ZQ

MODULE PROCEDURE IM_Q1

MODULE PROCEDURE IM_ZQ1

MODULE PROCEDURE IM_Q2

MODULE PROCEDURE IM_ZQ2

END INTERFACE

INTERFACE TO_ZM

MODULE PROCEDURE ZM_Q

```
MODULE PROCEDURE ZM2_Q
MODULE PROCEDURE ZM_ZQ
MODULE PROCEDURE ZM_Q1
MODULE PROCEDURE ZM_ZQ1
MODULE PROCEDURE ZM_Q2
MODULE PROCEDURE ZM_ZQ2
END INTERFACE
```

```
INTERFACE TO_QUAD
MODULE PROCEDURE FM_2QUAD
MODULE PROCEDURE IM_2QUAD
MODULE PROCEDURE ZM_2QUAD
MODULE PROCEDURE FM_2QUAD1
MODULE PROCEDURE IM_2QUAD1
MODULE PROCEDURE ZM_2QUAD1
MODULE PROCEDURE FM_2QUAD2
MODULE PROCEDURE IM_2QUAD2
MODULE PROCEDURE ZM_2QUAD2
END INTERFACE
```

```
INTERFACE TO_QUAD_Z
MODULE PROCEDURE FM_2QUADZ
MODULE PROCEDURE IM_2QUADZ
MODULE PROCEDURE ZM_2QUADZ
MODULE PROCEDURE FM_2QUADZ1
MODULE PROCEDURE IM_2QUADZ1
MODULE PROCEDURE ZM_2QUADZ1
MODULE PROCEDURE FM_2QUADZ2
MODULE PROCEDURE IM_2QUADZ2
MODULE PROCEDURE ZM_2QUADZ2
END INTERFACE
```

```
INTERFACE ASSIGNMENT (=)
MODULE PROCEDURE FMEQ_QFM
MODULE PROCEDURE FMEQ_QIM
MODULE PROCEDURE FMEQ_QZM
MODULE PROCEDURE FMEQ_ZQFM
MODULE PROCEDURE FMEQ_ZQIM
MODULE PROCEDURE FMEQ_ZQZM
MODULE PROCEDURE FMEQ_FMQ
MODULE PROCEDURE FMEQ_FMZQ
MODULE PROCEDURE FMEQ_IMQ
MODULE PROCEDURE FMEQ_IMZQ
MODULE PROCEDURE FMEQ_ZMQ
MODULE PROCEDURE FMEQ_ZMZQ
MODULE PROCEDURE FMEQ_FM1Q
MODULE PROCEDURE FMEQ_FM1ZQ
MODULE PROCEDURE FMEQ_Q1FM
MODULE PROCEDURE FMEQ_ZQ1FM
MODULE PROCEDURE FMEQ_FM1Q1
MODULE PROCEDURE FMEQ_FM1ZQ1
MODULE PROCEDURE FMEQ_Q1FM1
MODULE PROCEDURE FMEQ_ZQ1FM1
MODULE PROCEDURE FMEQ_IM1Q
MODULE PROCEDURE FMEQ_IM1ZQ
MODULE PROCEDURE FMEQ_Q1IM
MODULE PROCEDURE FMEQ_ZQ1IM
MODULE PROCEDURE FMEQ_IM1Q1
```

```
MODULE PROCEDURE FMEQ_IM1ZQ1
MODULE PROCEDURE FMEQ_Q1IM1
MODULE PROCEDURE FMEQ_ZQ1IM1
MODULE PROCEDURE FMEQ_ZM1Q
MODULE PROCEDURE FMEQ_ZM1ZQ
MODULE PROCEDURE FMEQ_Q1ZM
MODULE PROCEDURE FMEQ_ZQ1ZM
MODULE PROCEDURE FMEQ_ZM1Q1
MODULE PROCEDURE FMEQ_ZM1ZQ1
MODULE PROCEDURE FMEQ_Q1ZM1
MODULE PROCEDURE FMEQ_ZQ1ZM1
MODULE PROCEDURE FMEQ_FM2Q
MODULE PROCEDURE FMEQ_FM2ZQ
MODULE PROCEDURE FMEQ_Q2FM
MODULE PROCEDURE FMEQ_ZQ2FM
MODULE PROCEDURE FMEQ_FM2Q2
MODULE PROCEDURE FMEQ_FM2ZQ2
MODULE PROCEDURE FMEQ_Q2FM2
MODULE PROCEDURE FMEQ_ZQ2FM2
MODULE PROCEDURE FMEQ_IM2Q
MODULE PROCEDURE FMEQ_IM2ZQ
MODULE PROCEDURE FMEQ_Q2IM
MODULE PROCEDURE FMEQ_ZQ2IM
MODULE PROCEDURE FMEQ_IM2Q2
MODULE PROCEDURE FMEQ_IM2ZQ2
MODULE PROCEDURE FMEQ_Q2IM2
MODULE PROCEDURE FMEQ_ZQ2IM2
MODULE PROCEDURE FMEQ_ZM2Q
MODULE PROCEDURE FMEQ_ZM2ZQ
MODULE PROCEDURE FMEQ_Q2ZM
MODULE PROCEDURE FMEQ_ZQ2ZM
MODULE PROCEDURE FMEQ_ZM2Q2
MODULE PROCEDURE FMEQ_ZM2ZQ2
MODULE PROCEDURE FMEQ_Q2ZM2
MODULE PROCEDURE FMEQ_ZQ2ZM2
```

END INTERFACE

INTERFACE OPERATOR (==)

```
MODULE PROCEDURE FMLEQ_QFM
MODULE PROCEDURE FMLEQ_QIM
MODULE PROCEDURE FMLEQ_QZM
MODULE PROCEDURE FMLEQ_ZQFM
MODULE PROCEDURE FMLEQ_ZQIM
MODULE PROCEDURE FMLEQ_ZQZM
MODULE PROCEDURE FMLEQ_FMQ
MODULE PROCEDURE FMLEQ_FMZQ
MODULE PROCEDURE FMLEQ_IMQ
MODULE PROCEDURE FMLEQ_IMZQ
MODULE PROCEDURE FMLEQ_ZMQ
MODULE PROCEDURE FMLEQ_ZMZQ
```

END INTERFACE

INTERFACE OPERATOR (/=)

```
MODULE PROCEDURE FMLNE_QFM
MODULE PROCEDURE FMLNE_QIM
MODULE PROCEDURE FMLNE_QZM
MODULE PROCEDURE FMLNE_ZQFM
MODULE PROCEDURE FMLNE_ZQIM
```

```
MODULE PROCEDURE FMLNE_ZQZM
MODULE PROCEDURE FMLNE_FMQ
MODULE PROCEDURE FMLNE_FMZQ
MODULE PROCEDURE FMLNE_IMQ
MODULE PROCEDURE FMLNE_IMZQ
MODULE PROCEDURE FMLNE_ZMQ
MODULE PROCEDURE FMLNE_ZMZQ
END INTERFACE
```

```
INTERFACE OPERATOR (>)
MODULE PROCEDURE FMLGT_QFM
MODULE PROCEDURE FMLGT_QIM
MODULE PROCEDURE FMLGT_FMQ
MODULE PROCEDURE FMLGT_IMQ
END INTERFACE
```

```
INTERFACE OPERATOR (>=)
MODULE PROCEDURE FMLGE_QFM
MODULE PROCEDURE FMLGE_QIM
MODULE PROCEDURE FMLGE_FMQ
MODULE PROCEDURE FMLGE_IMQ
END INTERFACE
```

```
INTERFACE OPERATOR (<)
MODULE PROCEDURE FMLLT_QFM
MODULE PROCEDURE FMLLT_QIM
MODULE PROCEDURE FMLLT_FMQ
MODULE PROCEDURE FMLLT_IMQ
END INTERFACE
```

```
INTERFACE OPERATOR (<=)
MODULE PROCEDURE FMLLE_QFM
MODULE PROCEDURE FMLLE_QIM
MODULE PROCEDURE FMLLE_FMQ
MODULE PROCEDURE FMLLE_IMQ
END INTERFACE
```

```
INTERFACE OPERATOR (+)
MODULE PROCEDURE FMADD_QFM
MODULE PROCEDURE FMADD_QIM
MODULE PROCEDURE FMADD_QZM
MODULE PROCEDURE FMADD_ZQFM
MODULE PROCEDURE FMADD_ZQIM
MODULE PROCEDURE FMADD_ZQZM
MODULE PROCEDURE FMADD_FMQ
MODULE PROCEDURE FMADD_FMZQ
MODULE PROCEDURE FMADD_IMQ
MODULE PROCEDURE FMADD_IMZQ
MODULE PROCEDURE FMADD_ZMQ
MODULE PROCEDURE FMADD_ZMZQ
MODULE PROCEDURE FMADD_QFM1
MODULE PROCEDURE FMADD_ZQFM1
MODULE PROCEDURE FMADD_FMQ1
MODULE PROCEDURE FMADD_FMZQ1
MODULE PROCEDURE FMADD_FM1Q
MODULE PROCEDURE FMADD_FM1ZQ
MODULE PROCEDURE FMADD_Q1FM
MODULE PROCEDURE FMADD_ZQ1FM
```

MODULE PROCEDURE FMADD_Q1FM1
MODULE PROCEDURE FMADD_ZQ1FM1
MODULE PROCEDURE FMADD_FM1Q1
MODULE PROCEDURE FMADD_FM1ZQ1
MODULE PROCEDURE FMADD_QIM1
MODULE PROCEDURE FMADD_ZQIM1
MODULE PROCEDURE FMADD_IMQ1
MODULE PROCEDURE FMADD_IMZQ1
MODULE PROCEDURE FMADD_IM1Q
MODULE PROCEDURE FMADD_IM1ZQ
MODULE PROCEDURE FMADD_Q1IM
MODULE PROCEDURE FMADD_ZQ1IM
MODULE PROCEDURE FMADD_Q1IM1
MODULE PROCEDURE FMADD_ZQ1IM1
MODULE PROCEDURE FMADD_IM1Q1
MODULE PROCEDURE FMADD_IM1ZQ1
MODULE PROCEDURE FMADD_QZM1
MODULE PROCEDURE FMADD_ZQZM1
MODULE PROCEDURE FMADD_ZMQ1
MODULE PROCEDURE FMADD_ZMZQ1
MODULE PROCEDURE FMADD_ZM1Q
MODULE PROCEDURE FMADD_ZM1ZQ
MODULE PROCEDURE FMADD_Q1ZM
MODULE PROCEDURE FMADD_ZQ1ZM
MODULE PROCEDURE FMADD_Q1ZM1
MODULE PROCEDURE FMADD_ZQ1ZM1
MODULE PROCEDURE FMADD_ZM1Q1
MODULE PROCEDURE FMADD_ZM1ZQ1
MODULE PROCEDURE FMADD_QFM2
MODULE PROCEDURE FMADD_ZQFM2
MODULE PROCEDURE FMADD_FMQ2
MODULE PROCEDURE FMADD_FMZQ2
MODULE PROCEDURE FMADD_FM2Q
MODULE PROCEDURE FMADD_FM2ZQ
MODULE PROCEDURE FMADD_Q2FM
MODULE PROCEDURE FMADD_ZQ2FM
MODULE PROCEDURE FMADD_Q2FM2
MODULE PROCEDURE FMADD_ZQ2FM2
MODULE PROCEDURE FMADD_FM2Q2
MODULE PROCEDURE FMADD_FM2ZQ2
MODULE PROCEDURE FMADD_QIM2
MODULE PROCEDURE FMADD_ZQIM2
MODULE PROCEDURE FMADD_IMQ2
MODULE PROCEDURE FMADD_IMZQ2
MODULE PROCEDURE FMADD_IM2Q
MODULE PROCEDURE FMADD_IM2ZQ
MODULE PROCEDURE FMADD_Q2IM
MODULE PROCEDURE FMADD_ZQ2IM
MODULE PROCEDURE FMADD_Q2IM2
MODULE PROCEDURE FMADD_ZQ2IM2
MODULE PROCEDURE FMADD_IM2Q2
MODULE PROCEDURE FMADD_IM2ZQ2
MODULE PROCEDURE FMADD_QZM2
MODULE PROCEDURE FMADD_ZQZM2
MODULE PROCEDURE FMADD_ZMQ2
MODULE PROCEDURE FMADD_ZMZQ2
MODULE PROCEDURE FMADD_ZM2Q
MODULE PROCEDURE FMADD_ZM2ZQ

```
MODULE PROCEDURE FMADD_Q2ZM
MODULE PROCEDURE FMADD_ZQ2ZM
MODULE PROCEDURE FMADD_Q2ZM2
MODULE PROCEDURE FMADD_ZQ2ZM2
MODULE PROCEDURE FMADD_ZM2Q2
MODULE PROCEDURE FMADD_ZM2ZQ2
END INTERFACE
```

```
INTERFACE OPERATOR (-)
```

```
MODULE PROCEDURE FMSUB_QFM
MODULE PROCEDURE FMSUB_QIM
MODULE PROCEDURE FMSUB_QZM
MODULE PROCEDURE FMSUB_ZQFM
MODULE PROCEDURE FMSUB_ZQIM
MODULE PROCEDURE FMSUB_ZQZM
MODULE PROCEDURE FMSUB_FMQ
MODULE PROCEDURE FMSUB_FMZQ
MODULE PROCEDURE FMSUB_IMQ
MODULE PROCEDURE FMSUB_IMZQ
MODULE PROCEDURE FMSUB_ZMQ
MODULE PROCEDURE FMSUB_ZMZQ
MODULE PROCEDURE FMSUB_QFM1
MODULE PROCEDURE FMSUB_ZQFM1
MODULE PROCEDURE FMSUB_FMQ1
MODULE PROCEDURE FMSUB_FMZQ1
MODULE PROCEDURE FMSUB_FM1Q
MODULE PROCEDURE FMSUB_FM1ZQ
MODULE PROCEDURE FMSUB_Q1FM
MODULE PROCEDURE FMSUB_ZQ1FM
MODULE PROCEDURE FMSUB_Q1FM1
MODULE PROCEDURE FMSUB_ZQ1FM1
MODULE PROCEDURE FMSUB_FM1Q1
MODULE PROCEDURE FMSUB_FM1ZQ1
MODULE PROCEDURE FMSUB_QIM1
MODULE PROCEDURE FMSUB_ZQIM1
MODULE PROCEDURE FMSUB_IMQ1
MODULE PROCEDURE FMSUB_IMZQ1
MODULE PROCEDURE FMSUB_IM1Q
MODULE PROCEDURE FMSUB_IM1ZQ
MODULE PROCEDURE FMSUB_Q1IM
MODULE PROCEDURE FMSUB_ZQ1IM
MODULE PROCEDURE FMSUB_Q1IM1
MODULE PROCEDURE FMSUB_ZQ1IM1
MODULE PROCEDURE FMSUB_IM1Q1
MODULE PROCEDURE FMSUB_IM1ZQ1
MODULE PROCEDURE FMSUB_QZM1
MODULE PROCEDURE FMSUB_ZQZM1
MODULE PROCEDURE FMSUB_ZMQ1
MODULE PROCEDURE FMSUB_ZMZQ1
MODULE PROCEDURE FMSUB_ZM1Q
MODULE PROCEDURE FMSUB_ZM1ZQ
MODULE PROCEDURE FMSUB_Q1ZM
MODULE PROCEDURE FMSUB_ZQ1ZM
MODULE PROCEDURE FMSUB_Q1ZM1
MODULE PROCEDURE FMSUB_ZQ1ZM1
MODULE PROCEDURE FMSUB_ZM1Q1
MODULE PROCEDURE FMSUB_ZM1ZQ1
MODULE PROCEDURE FMSUB_QFM2
```

MODULE PROCEDURE FMSUB_ZQFM2
MODULE PROCEDURE FMSUB_FM2Q2
MODULE PROCEDURE FMSUB_FM2Q
MODULE PROCEDURE FMSUB_FM2ZQ
MODULE PROCEDURE FMSUB_Q2FM
MODULE PROCEDURE FMSUB_ZQ2FM
MODULE PROCEDURE FMSUB_Q2FM2
MODULE PROCEDURE FMSUB_ZQ2FM2
MODULE PROCEDURE FMSUB_FM2Q2
MODULE PROCEDURE FMSUB_FM2ZQ2
MODULE PROCEDURE FMSUB_QIM2
MODULE PROCEDURE FMSUB_ZQIM2
MODULE PROCEDURE FMSUB_IM2Q2
MODULE PROCEDURE FMSUB_IM2Q
MODULE PROCEDURE FMSUB_IM2ZQ
MODULE PROCEDURE FMSUB_Q2IM
MODULE PROCEDURE FMSUB_ZQ2IM
MODULE PROCEDURE FMSUB_Q2IM2
MODULE PROCEDURE FMSUB_ZQ2IM2
MODULE PROCEDURE FMSUB_IM2Q2
MODULE PROCEDURE FMSUB_IM2ZQ2
MODULE PROCEDURE FMSUB_QZM2
MODULE PROCEDURE FMSUB_ZQZM2
MODULE PROCEDURE FMSUB_ZMQ2
MODULE PROCEDURE FMSUB_ZMZQ2
MODULE PROCEDURE FMSUB_ZM2Q
MODULE PROCEDURE FMSUB_ZM2ZQ
MODULE PROCEDURE FMSUB_Q2ZM
MODULE PROCEDURE FMSUB_ZQ2ZM
MODULE PROCEDURE FMSUB_Q2ZM2
MODULE PROCEDURE FMSUB_ZQ2ZM2
MODULE PROCEDURE FMSUB_ZM2Q2
MODULE PROCEDURE FMSUB_ZM2ZQ2

END INTERFACE

INTERFACE OPERATOR (*)

MODULE PROCEDURE FMMPY_QFM
MODULE PROCEDURE FMMPY_QIM
MODULE PROCEDURE FMMPY_QZM
MODULE PROCEDURE FMMPY_ZQFM
MODULE PROCEDURE FMMPY_ZQIM
MODULE PROCEDURE FMMPY_ZQZM
MODULE PROCEDURE FMMPY_FM2Q
MODULE PROCEDURE FMMPY_FM2Q2
MODULE PROCEDURE FMMPY_IM2Q
MODULE PROCEDURE FMMPY_IM2Q2
MODULE PROCEDURE FMMPY_ZM2Q
MODULE PROCEDURE FMMPY_ZM2Q2
MODULE PROCEDURE FMMPY_QFM1
MODULE PROCEDURE FMMPY_ZQFM1
MODULE PROCEDURE FMMPY_FM2Q1
MODULE PROCEDURE FMMPY_FM2Q12
MODULE PROCEDURE FMMPY_FM1Q
MODULE PROCEDURE FMMPY_FM1Q2
MODULE PROCEDURE FMMPY_Q1FM
MODULE PROCEDURE FMMPY_ZQ1FM

MODULE PROCEDURE FMMPY_Q1FM1
MODULE PROCEDURE FMMPY_ZQ1FM1
MODULE PROCEDURE FMMPY_FM1Q1
MODULE PROCEDURE FMMPY_FM1ZQ1
MODULE PROCEDURE FMMPY_QIM1
MODULE PROCEDURE FMMPY_ZQIM1
MODULE PROCEDURE FMMPY_IMQ1
MODULE PROCEDURE FMMPY_IMZQ1
MODULE PROCEDURE FMMPY_IM1Q
MODULE PROCEDURE FMMPY_IM1ZQ
MODULE PROCEDURE FMMPY_Q1IM
MODULE PROCEDURE FMMPY_ZQ1IM
MODULE PROCEDURE FMMPY_Q1IM1
MODULE PROCEDURE FMMPY_ZQ1IM1
MODULE PROCEDURE FMMPY_IM1Q1
MODULE PROCEDURE FMMPY_IM1ZQ1
MODULE PROCEDURE FMMPY_QZM1
MODULE PROCEDURE FMMPY_ZQZM1
MODULE PROCEDURE FMMPY_ZMQ1
MODULE PROCEDURE FMMPY_ZMZQ1
MODULE PROCEDURE FMMPY_ZM1Q
MODULE PROCEDURE FMMPY_ZM1ZQ
MODULE PROCEDURE FMMPY_Q1ZM
MODULE PROCEDURE FMMPY_ZQ1ZM
MODULE PROCEDURE FMMPY_Q1ZM1
MODULE PROCEDURE FMMPY_ZQ1ZM1
MODULE PROCEDURE FMMPY_ZM1Q1
MODULE PROCEDURE FMMPY_ZM1ZQ1
MODULE PROCEDURE FMMPY_QFM2
MODULE PROCEDURE FMMPY_ZQFM2
MODULE PROCEDURE FMMPY_FMQ2
MODULE PROCEDURE FMMPY_FMZQ2
MODULE PROCEDURE FMMPY_FM2Q
MODULE PROCEDURE FMMPY_FM2ZQ
MODULE PROCEDURE FMMPY_Q2FM
MODULE PROCEDURE FMMPY_ZQ2FM
MODULE PROCEDURE FMMPY_Q2FM2
MODULE PROCEDURE FMMPY_ZQ2FM2
MODULE PROCEDURE FMMPY_FM2Q2
MODULE PROCEDURE FMMPY_FM2ZQ2
MODULE PROCEDURE FMMPY_QIM2
MODULE PROCEDURE FMMPY_ZQIM2
MODULE PROCEDURE FMMPY_IMQ2
MODULE PROCEDURE FMMPY_IMZQ2
MODULE PROCEDURE FMMPY_IM2Q
MODULE PROCEDURE FMMPY_IM2ZQ
MODULE PROCEDURE FMMPY_Q2IM
MODULE PROCEDURE FMMPY_ZQ2IM
MODULE PROCEDURE FMMPY_Q2IM2
MODULE PROCEDURE FMMPY_ZQ2IM2
MODULE PROCEDURE FMMPY_IM2Q2
MODULE PROCEDURE FMMPY_IM2ZQ2
MODULE PROCEDURE FMMPY_QZM2
MODULE PROCEDURE FMMPY_ZQZM2
MODULE PROCEDURE FMMPY_ZMQ2
MODULE PROCEDURE FMMPY_ZMZQ2
MODULE PROCEDURE FMMPY_ZM2Q
MODULE PROCEDURE FMMPY_ZM2ZQ


```
MODULE PROCEDURE FMMPY_Q2ZM
MODULE PROCEDURE FMMPY_ZQ2ZM
MODULE PROCEDURE FMMPY_Q2ZM2
MODULE PROCEDURE FMMPY_ZQ2ZM2
MODULE PROCEDURE FMMPY_ZM2Q2
MODULE PROCEDURE FMMPY_ZM2ZQ2
END INTERFACE
```

```
INTERFACE OPERATOR (/)
```

```
MODULE PROCEDURE FMDIV_QFM
MODULE PROCEDURE FMDIV_QIM
MODULE PROCEDURE FMDIV_QZM
MODULE PROCEDURE FMDIV_ZQFM
MODULE PROCEDURE FMDIV_ZQIM
MODULE PROCEDURE FMDIV_ZQZM
MODULE PROCEDURE FMDIV_FMQ
MODULE PROCEDURE FMDIV_FMZQ
MODULE PROCEDURE FMDIV_IMQ
MODULE PROCEDURE FMDIV_IMZQ
MODULE PROCEDURE FMDIV_ZMQ
MODULE PROCEDURE FMDIV_ZMZQ
MODULE PROCEDURE FMDIV_QFM1
MODULE PROCEDURE FMDIV_ZQFM1
MODULE PROCEDURE FMDIV_FMQ1
MODULE PROCEDURE FMDIV_FMZQ1
MODULE PROCEDURE FMDIV_FM1Q
MODULE PROCEDURE FMDIV_FM1ZQ
MODULE PROCEDURE FMDIV_Q1FM
MODULE PROCEDURE FMDIV_ZQ1FM
MODULE PROCEDURE FMDIV_Q1FM1
MODULE PROCEDURE FMDIV_ZQ1FM1
MODULE PROCEDURE FMDIV_FM1Q1
MODULE PROCEDURE FMDIV_FM1ZQ1
MODULE PROCEDURE FMDIV_QIM1
MODULE PROCEDURE FMDIV_ZQIM1
MODULE PROCEDURE FMDIV_IMQ1
MODULE PROCEDURE FMDIV_IMZQ1
MODULE PROCEDURE FMDIV_IM1Q
MODULE PROCEDURE FMDIV_IM1ZQ
MODULE PROCEDURE FMDIV_Q1IM
MODULE PROCEDURE FMDIV_ZQ1IM
MODULE PROCEDURE FMDIV_Q1IM1
MODULE PROCEDURE FMDIV_ZQ1IM1
MODULE PROCEDURE FMDIV_IM1Q1
MODULE PROCEDURE FMDIV_IM1ZQ1
MODULE PROCEDURE FMDIV_QZM1
MODULE PROCEDURE FMDIV_ZQZM1
MODULE PROCEDURE FMDIV_ZMQ1
MODULE PROCEDURE FMDIV_ZMZQ1
MODULE PROCEDURE FMDIV_ZM1Q
MODULE PROCEDURE FMDIV_ZM1ZQ
MODULE PROCEDURE FMDIV_Q1ZM
MODULE PROCEDURE FMDIV_ZQ1ZM
MODULE PROCEDURE FMDIV_Q1ZM1
MODULE PROCEDURE FMDIV_ZQ1ZM1
MODULE PROCEDURE FMDIV_ZM1Q1
MODULE PROCEDURE FMDIV_ZM1ZQ1
MODULE PROCEDURE FMDIV_QFM2
```

```
MODULE PROCEDURE FMDIV_ZQFM2
MODULE PROCEDURE FMDIV_FM2Q2
MODULE PROCEDURE FMDIV_FM2Q
MODULE PROCEDURE FMDIV_FM2ZQ
MODULE PROCEDURE FMDIV_Q2FM
MODULE PROCEDURE FMDIV_Q2FM2
MODULE PROCEDURE FMDIV_Q2FM2
MODULE PROCEDURE FMDIV_Q2FM2
MODULE PROCEDURE FMDIV_FM2Q2
MODULE PROCEDURE FMDIV_FM2ZQ2
MODULE PROCEDURE FMDIV_QIM2
MODULE PROCEDURE FMDIV_ZQIM2
MODULE PROCEDURE FMDIV_IM2Q2
MODULE PROCEDURE FMDIV_IM2Q2
MODULE PROCEDURE FMDIV_IM2Q
MODULE PROCEDURE FMDIV_IM2ZQ
MODULE PROCEDURE FMDIV_Q2IM
MODULE PROCEDURE FMDIV_ZQ2IM
MODULE PROCEDURE FMDIV_Q2IM2
MODULE PROCEDURE FMDIV_ZQ2IM2
MODULE PROCEDURE FMDIV_IM2Q2
MODULE PROCEDURE FMDIV_IM2ZQ2
MODULE PROCEDURE FMDIV_QZM2
MODULE PROCEDURE FMDIV_ZQZM2
MODULE PROCEDURE FMDIV_ZMQ2
MODULE PROCEDURE FMDIV_ZMZQ2
MODULE PROCEDURE FMDIV_ZM2Q
MODULE PROCEDURE FMDIV_ZM2ZQ
MODULE PROCEDURE FMDIV_Q2ZM
MODULE PROCEDURE FMDIV_ZQ2ZM
MODULE PROCEDURE FMDIV_Q2ZM2
MODULE PROCEDURE FMDIV_ZQ2ZM2
MODULE PROCEDURE FMDIV_ZM2Q2
MODULE PROCEDURE FMDIV_ZM2ZQ2
END INTERFACE
```

```
INTERFACE OPERATOR (**)
MODULE PROCEDURE FMPWR_QFM
MODULE PROCEDURE FMPWR_QIM
MODULE PROCEDURE FMPWR_QZM
MODULE PROCEDURE FMPWR_ZQFM
MODULE PROCEDURE FMPWR_ZQIM
MODULE PROCEDURE FMPWR_ZQZM
MODULE PROCEDURE FMPWR_FM2Q
MODULE PROCEDURE FMPWR_FM2Q
MODULE PROCEDURE FMPWR_IM2Q
MODULE PROCEDURE FMPWR_IM2Q
MODULE PROCEDURE FMPWR_ZM2Q
MODULE PROCEDURE FMPWR_ZM2Q
END INTERFACE
```

CONTAINS

```
SUBROUTINE FMQ2M(X,MA)
```

! Convert quadruple precision X to multiple precision MA.

```
USE FMVALS
IMPLICIT NONE
```

```
REAL (QUAD_FP) :: X
TYPE(MULTI) :: MA
REAL (QUAD_FP) :: F1,F2,Y,Y1,Y2,TWO20
INTEGER :: J,J1,J2,JD,JEXP,K,KEXP,L,NDSAVE
INTENT (IN) :: X
INTENT (INOUT) :: MA
TYPE(MULTI) :: MXY(5)
```

```
IF (.NOT. ALLOCATED(MA%MP)) THEN
  ALLOCATE(MA%MP(NDIG+2),STAT=K_STAT)
  IF (K_STAT /= 0) CALL FMDEFINE_ERROR
ELSE IF (SIZE(MA%MP) < NDIG+2) THEN
  DEALLOCATE(MA%MP)
  ALLOCATE(MA%MP(NDIG+2),STAT=K_STAT)
  IF (K_STAT /= 0) CALL FMDEFINE_ERROR
ENDIF
```

! Increase the working precision.

```
NDSAVE = NDIG
IF (NCALL == 1) THEN
  K = MAX(NGRD21,1)
  NDIG = MAX(NDIG+K,2)
ENDIF
```

```
IF (MBLOGS /= MBASE) CALL FMCONS
KFLAG = 0
```

! Special case for X = 0.

```
IF (X == 0) THEN
  DO J = 1, NDSAVE+1
    MA%MP(J+1) = 0
  ENDDO
  GO TO 240
ENDIF
```

! Check for X = + or - Infinity, or NaN. Return unknown if so.

```
IF (X > HUGE(X) .OR. X < -HUGE(X) .OR. (.NOT.(X == X))) THEN
  DO J = 2, NDSAVE
    MA%MP(J+2) = 0
  ENDDO
  KFLAG = -4
  MA%MP(2) = MUNKNO
  MA%MP(3) = 1
  MA%MP(1) = 1
  CALL FMWARN
  NDIG = NDSAVE
  RETURN
ENDIF
```

! Special case for MBASE = 2.

```
IF (MBASE == 2 .AND. RADIX(X) == 2) THEN
```

```

NDIG = MAX(NDIG,DIGITS(X))
Y = FRACTION(ABS(X))
CALL FMI2M(0,MXY(5))
DO J = 1, MIN(DIGITS(X),NDIG)
  Y = Y + Y
  MXY(5)%MP(J+2) = INT(Y)
  Y = Y - INT(Y)
ENDDO
MXY(5)%MP(2) = EXPONENT(X)
CALL FMEQU(MXY(5),MA,NDIG,NDSAVE)
GO TO 240
ENDIF

```

! Special case for MBASE = 10**L.

```

K = MBASE
L = 0
DO
  IF (MOD(K,10) == 0) THEN
    L = L + 1
    K = K/10
    IF (K == 1) EXIT
  ELSE
    L = 0
    EXIT
  ENDIF
ENDDO
IF (L > 0) THEN
  NDIG = MAX(NDIG,INT(DIGITS(X)*0.30103/L)+1)
  Y = FRACTION(ABS(X))
  CALL FMI2M(0,MXY(5))
  DO J = 1, NDIG

```

! Multiply by 10**L to get the next digit in base MBASE.
! To avoid any rounding errors in quad precision, do each multiply by 10 as
! one multiply by 8 and one by 2, and keep two integer and two fraction results.
! So 10*y is broken into 8*y + 2*y, since there will be no rounding with either
! term in quad precision on a binary machine.

```

  JD = 0
  DO K = 1, L
    Y1 = 8*Y
    Y2 = 2*Y
    J1 = Y1
    J2 = Y2
    F1 = Y1 - J1
    F2 = Y2 - J2
    JD = 10*JD + J1 + J2
    Y = F1 + F2
    IF (Y >= 1) THEN
      JD = JD + 1
      Y = Y - 1
    ENDIF
  ENDDO
  MXY(5)%MP(J+2) = JD
  IF (Y == 0) EXIT
ENDDO
K = INTMAX

```

```

IF (MAXINT/MBASE < K) K = MAXINT/MBASE
K = K/2
J2 = 1
JEXP = EXPONENT(X)
DO J = 1, ABS(JEXP)
  J2 = 2*J2
  IF (J2 >= K .OR. J == ABS(JEXP)) THEN
    IF (JEXP > 0) THEN
      CALL FMMPYI_R1(MXY(5),J2)
    ELSE
      CALL FMDIVI_R1(MXY(5),J2)
    ENDIF
    J2 = 1
  ENDF
ENDDO
CALL FMEQU(MXY(5),MA,NDIG,NDSAVE)
GO TO 240
ENDIF

```

```

Y = ABS(X)
TWO20 = 1048576.0D0

```

! If this power of two is not representable at the current base and precision, use a
! smaller one.

```

IF (INT(NDIG*ALOGM2) < 20) THEN
  K = INT(NDIG*ALOGM2)
  TWO20 = 1
  DO J = 1, K
    TWO20 = TWO20*2.0D0
  ENDDO
ENDIF

```

```

KEXP = 0
IF (Y > TWO20) THEN
130  Y = Y/TWO20
    KEXP = KEXP + 1
    IF (Y > TWO20) GO TO 130
ELSE IF (Y < 1) THEN
140  Y = Y*TWO20
    KEXP = KEXP - 1
    IF (Y < 1) GO TO 140
ENDIF

```

```

K = INT(TWO20)
CALL FMI2M(K,MXY(3))
K = INT(Y)
CALL FMI2M(K,MXY(1))
Y = (Y-DBLE(K))*TWO20
JEXP = 0

```

```

160 K = INT(Y)
CALL FMI2M(K,MXY(2))
CALL FMMPY_R1(MXY(1),MXY(3))
JEXP = JEXP + 1
CALL FMADD_R1(MXY(1),MXY(2))
Y = (Y-DBLE(K))*TWO20
IF (JEXP <= 1000 .AND. Y /= 0) GO TO 160

```

```

K = KEXP - JEXP
IF (K >= 0) THEN
  IF (K == 0) THEN
    CALL FMEQ(MXY(1),MXY(5))
  ELSE IF (K == 1) THEN
    CALL FMMPY(MXY(1),MXY(3),MXY(5))
  ELSE IF (K == 2) THEN
    CALL FMSQR(MXY(3),MXY(2))
    CALL FMMPY(MXY(1),MXY(2),MXY(5))
  ELSE
    CALL FMIPWR(MXY(3),K,MXY(2))
    CALL FMMPY(MXY(1),MXY(2),MXY(5))
  ENDIF
ELSE
  IF (K == -1) THEN
    CALL FMDIV(MXY(1),MXY(3),MXY(5))
  ELSE IF (K == -2) THEN
    CALL FMSQR(MXY(3),MXY(2))
    CALL FMDIV(MXY(1),MXY(2),MXY(5))
  ELSE
    CALL FMIPWR(MXY(3),-K,MXY(2))
    CALL FMDIV(MXY(1),MXY(2),MXY(5))
  ENDIF
ENDIF
CALL FMEQU(MXY(5),MA,NDIG,NDSAVE)

```

```

240 MA%MP(1) = 1
IF (X < 0.0 .AND. MA%MP(2) /= MUNKNO .AND. MA%MP(3) /= 0) MA%MP(1) = -1
NDIG = NDSAVE
RETURN
END SUBROUTINE FMQ2M

```

```

SUBROUTINE FMM2Q(MA,X)

```

! Convert multiple precision MA to quadruple precision X.

```

USE FMVALS
IMPLICIT NONE

TYPE(MULTI) :: MA
REAL (QUAD_FP) :: X

REAL (QUAD_FP) :: AQ(2),XQ(2),YQ(2),Y1(2),Y2(2),XBASE,PMAX,DLOGDP, &
  A1,A2,C,C1,C2,C21,C22,Q1,Q2,T,Z1,Z2
REAL (KIND(1.0D0)) :: MA1,MAS
INTEGER :: J,K,KWRNSV,NCASE
INTENT (IN) :: MA
INTENT (INOUT) :: X

```

! Check to see if MA is in range for quadruple precision.

```

IF (MBLOGS /= MBASE) CALL FMCONS
PMAX = HUGE(X) / 5
DLOGDP = LOG(PMAX)
MA1 = MA%MP(2)
NCASE = 0

```

```

IF (DBLE(MA%MP(2)-1)*DLOGMB > DLOGDP) THEN
  KFLAG = -4
  X = -1.01*(HUGE(X)/3.0)
  CALL FMWARN
  RETURN
ELSE IF (DBLE(MA%MP(2)+1)*DLOGMB > DLOGDP) THEN
  MA1 = MA1 - 2
  NCASE = 1
ELSE IF (DBLE(MA%MP(2)+1)*DLOGMB < -DLOGDP) THEN
  KFLAG = -10
  X = 0
  CALL FMWARN
  RETURN
ELSE IF (DBLE(MA%MP(2)-1)*DLOGMB < -DLOGDP) THEN
  MA1 = MA1 + 2
  NCASE = 2
ENDIF

```

! Try FMMI first so that small integers will be converted quickly.

```

KWRNSV = KWARN
KWARN = 0
CALL FMMI(MA, J)
KWARN = KWRNSV
IF (KFLAG == 0) THEN
  X = J
  RETURN
ENDIF
KFLAG = 0

```

!
!
! General case.
! In order to get the correctly rounded X, the arithmetic for computing X is done
! with twice quadruple precision using the arrays of length 2.

```

MAS = MA%MP(1)
XBASE = MBASE
XQ = (/ 0 , 0 /)
YQ = (/ 1 , 0 /)
C = RADIX(X)
K = DIGITS(X) - DIGITS(X)/2
C = C ** K
K = (LOG(DBLE(RADIX(X)))/DLOGMB)*DIGITS(X) + NGRD52
DO J = 2, MIN(K+1, NDIG+1)
  Z1 = YQ(1) / XBASE
  T = XBASE*C
  A1 = (XBASE - T) + T
  A2 = XBASE - A1
  T = Z1*C
  C1 = (Z1 - T) + T
  C2 = Z1 - C1
  T = C2*C
  C21 = (C2 - T) + T
  C22 = C2 - C21
  Q1 = XBASE*Z1
  Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2 + C22*A2
  Z2 = (((YQ(1)-Q1) - Q2) + YQ(2)) / XBASE
  YQ(1) = Z1 + Z2
  YQ(2) = (Z1-YQ(1)) + Z2

```

```

T = YQ(1)*C
A1 = (YQ(1) - T) + T
A2 = YQ(1) - A1
T = DBLE(MA%MP(J+1))*C
C1 = (DBLE(MA%MP(J+1)) - T) + T
C2 = DBLE(MA%MP(J+1)) - C1
T = C2*C
C21 = (C2 - T) + T
C22 = C2 - C21
Q1 = YQ(1)*DBLE(MA%MP(J+1))
Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2 + C22*A2
Z2 = YQ(2)*DBLE(MA%MP(J+1)) + Q2
AQ(1) = Q1 + Z2
AQ(2) = (Q1-AQ(1)) + Z2
Z1 = XQ(1) + AQ(1)
Q1 = XQ(1) - Z1
Z2 = (((Q1+AQ(1)) + (XQ(1)-(Q1+Z1))) + XQ(2)) + AQ(2)
XQ(1) = Z1 + Z2
XQ(2) = (Z1-XQ(1)) + Z2

```

ENDDO

```

Y1 = (/ XBASE , Q_ZERO /)
K = ABS(MA1)
IF (MOD(K,2) == 0) THEN
  Y2 = (/ 1 , 0 /)
ELSE
  Y2 = (/ XBASE , Q_ZERO /)
ENDIF

```

120 K = K/2

```

T = Y1(1)*C
A1 = (Y1(1) - T) + T
A2 = Y1(1) - A1
T = Y1(1)*C
C1 = (Y1(1) - T) + T
C2 = Y1(1) - C1
T = C2*C
C21 = (C2 - T) + T
C22 = C2 - C21
Q1 = Y1(1)*Y1(1)
Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2 + C22*A2
Z2 = ((Y1(1) + Y1(2))*Y1(2) + Y1(2)*Y1(1)) + Q2
Y1(1) = Q1 + Z2
Y1(2) = (Q1-Y1(1)) + Z2
IF (MOD(K,2) == 1) THEN
  T = Y1(1)*C
  A1 = (Y1(1) - T) + T
  A2 = Y1(1) - A1
  T = Y2(1)*C
  C1 = (Y2(1) - T) + T
  C2 = Y2(1) - C1
  T = C2*C
  C21 = (C2 - T) + T
  C22 = C2 - C21
  Q1 = Y1(1)*Y2(1)
  Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2 + C22*A2
  Z2 = ((Y1(1) + Y1(2))*Y2(2) + Y1(2)*Y2(1)) + Q2
  Y2(1) = Q1 + Z2

```



```

      Y2(2) = (Q1-Y2(1)) + Z2
ENDIF
IF (K > 1) GO TO 120

IF (MA1 < 0) THEN
  Z1 = XQ(1) / Y2(1)
  T = Y2(1)*C
  A1 = (Y2(1) - T) + T
  A2 = Y2(1) - A1
  T = Z1*C
  C1 = (Z1 - T) + T
  C2 = Z1 - C1
  T = C2*C
  C21 = (C2 - T) + T
  C22 = C2 - C21
  Q1 = Y2(1)*Z1
  Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2) + C22*A2
  Z2 = (((XQ(1)-Q1) - Q2) + XQ(2)) - Z1*Y2(2)) / (Y2(1) + Y2(2))
  AQ(1) = Z1 + Z2
  AQ(2) = (Z1-AQ(1)) + Z2

```

```

ELSE
  T = XQ(1)*C
  A1 = (XQ(1) - T) + T
  A2 = XQ(1) - A1
  T = Y2(1)*C
  C1 = (Y2(1) - T) + T
  C2 = Y2(1) - C1
  T = C2*C
  C21 = (C2 - T) + T
  C22 = C2 - C21
  Q1 = XQ(1)*Y2(1)
  Q2 = (((A1*C1 - Q1) + A1*C2) + C1*A2) + C21*A2) + C22*A2
  Z2 = ((XQ(1) + XQ(2))*Y2(2) + XQ(2)*Y2(1)) + Q2
  AQ(1) = Q1 + Z2
  AQ(2) = (Q1-AQ(1)) + Z2

```

```

ENDIF

X = AQ(1) + AQ(2)

```

```

IF (MAS < 0) X = -X

```

! Check the result if it is near overflow or underflow.

```

IF (NCASE == 1) THEN
  IF (X <= PMAX/(XBASE*XBASE)) THEN
    X = X*XBASE*XBASE
  ELSE
    KFLAG = -4
    X = -1.01*(HUGE(X)/3.0)
    CALL FMWARN
  ENDIF
ELSE IF (NCASE == 2) THEN
  IF (X >= (1/PMAX)*XBASE*XBASE) THEN
    X = X/(XBASE*XBASE)
  ELSE
    KFLAG = -10
    X = 0
    CALL FMWARN
  ENDIF

```

```
ENDIF
ENDIF
RETURN
END SUBROUTINE FMM2Q
```

```
SUBROUTINE IMM2Q(MA,X)
```

```
! X = MA
```

```
! Convert an IM number to quadruple precision.
```

```
USE FMVALS
IMPLICIT NONE
```

```
TYPE(MULTI) :: MA
REAL (QUAD_FP) :: X
```

```
INTEGER :: ND2,NDSAVE
INTENT (IN) :: MA
INTENT (INOUT) :: X
```

```
NDSAVE = NDIG
NDIG = MAX(2,INT(MA%MP(2)))
ND2 = 2 - LOG(EPSILON(Q_ONE))/DLOGMB
IF (NDIG >= ND2) NDIG = ND2
```

```
CALL FMM2Q(MA,X)
```

```
NDIG = NDSAVE
RETURN
END SUBROUTINE IMM2Q
```

```
FUNCTION FM_Q(D)      RESULT (RETURN_VALUE)
USE FMVALS
IMPLICIT NONE
TYPE (FM) :: RETURN_VALUE
REAL (QUAD_FP) :: D
INTENT (IN) :: D
CALL FMQ2M(D,RETURN_VALUE%MFM)
END FUNCTION FM_Q
```

```
FUNCTION FM_ZQ(C)      RESULT (RETURN_VALUE)
USE FMVALS
IMPLICIT NONE
TYPE (FM) :: RETURN_VALUE
COMPLEX (QUAD_FP) :: C
INTENT (IN) :: C
CALL FMQ2M(REAL(C,QUAD_FP),RETURN_VALUE%MFM)
END FUNCTION FM_ZQ
```

```
FUNCTION FM_Q1(D)      RESULT (RETURN_VALUE)
USE FMVALS
IMPLICIT NONE
REAL (QUAD_FP), DIMENSION(:) :: D
TYPE (FM), DIMENSION(SIZE(D)) :: RETURN_VALUE
INTEGER :: J,N
```

```

INTENT (IN) :: D
N = SIZE(D)
DO J = 1, N
    CALL FMQ2M(D(J),RETURN_VALUE(J)%MFM)
ENDDO
END FUNCTION FM_Q1

FUNCTION FM_ZQ1(C)    RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    COMPLEX (QUAD_FP), DIMENSION(:) :: C
    TYPE (FM), DIMENSION(SIZE(C)) :: RETURN_VALUE
    INTEGER :: J,N
    INTENT (IN) :: C
    N = SIZE(C)
    DO J = 1, N
        CALL FMQ2M(REAL(C(J),QUAD_FP),RETURN_VALUE(J)%MFM)
    ENDDO
END FUNCTION FM_ZQ1

FUNCTION FM_Q2(D)    RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    REAL (QUAD_FP), DIMENSION(:,:) :: D
    TYPE (FM), DIMENSION(SIZE(D,DIM=1),SIZE(D,DIM=2)) :: RETURN_VALUE
    INTEGER :: J,K
    INTENT (IN) :: D
    DO J = 1, SIZE(D,DIM=1)
        DO K = 1, SIZE(D,DIM=2)
            CALL FMQ2M(D(J,K),RETURN_VALUE(J,K)%MFM)
        ENDDO
    ENDDO
END FUNCTION FM_Q2

FUNCTION FM_ZQ2(C)    RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    COMPLEX (QUAD_FP), DIMENSION(:,:) :: C
    TYPE (FM), DIMENSION(SIZE(C,DIM=1),SIZE(C,DIM=2)) :: RETURN_VALUE
    INTEGER :: J,K
    INTENT (IN) :: C
    DO J = 1, SIZE(C,DIM=1)
        DO K = 1, SIZE(C,DIM=2)
            CALL FMQ2M(REAL(C(J,K),QUAD_FP),RETURN_VALUE(J,K)%MFM)
        ENDDO
    ENDDO
END FUNCTION FM_ZQ2

FUNCTION IM_Q(D)    RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    TYPE (IM) :: RETURN_VALUE
    REAL (QUAD_FP) :: D
    CHARACTER(50) :: ST
    INTEGER :: IVAL
    INTENT (IN) :: D
    IF (ABS(D) < HUGE(1)) THEN
        IVAL = INT(D)
    
```

```

        CALL IMI2M(IVAL,RETURN_VALUE%MIM)
ELSE
    WRITE (ST, '(E50.39)') D
    CALL IMST2M(ST,RETURN_VALUE%MIM)
ENDIF
END FUNCTION IM_Q

```

```

FUNCTION IM_ZQ(C)      RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    TYPE (IM) :: RETURN_VALUE
    COMPLEX (QUAD_FP) :: C
    REAL (QUAD_FP) :: D
    CHARACTER(50) :: ST
    INTEGER :: IVAL
    INTENT (IN) :: C
    D = REAL(C,QUAD_FP)
    IF (ABS(D) < HUGE(1)) THEN
        IVAL = INT(D)
        CALL IMI2M(IVAL,RETURN_VALUE%MIM)
    ELSE
        WRITE (ST, '(E50.39)') D
        CALL IMST2M(ST,RETURN_VALUE%MIM)
    ENDIF
END FUNCTION IM_ZQ

```

```

FUNCTION IM_Q1(D)      RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    REAL (QUAD_FP), DIMENSION(:) :: D
    TYPE (IM), DIMENSION(SIZE(D)) :: RETURN_VALUE
    CHARACTER(50) :: ST
    INTEGER :: IVAL,J,N
    INTENT (IN) :: D
    N = SIZE(D)
    DO J = 1, N
        IF (ABS(D(J)) < HUGE(1)) THEN
            IVAL = INT(D(J))
            CALL IMI2M(IVAL,RETURN_VALUE(J)%MIM)
        ELSE
            WRITE (ST, '(E50.39)') D(J)
            CALL IMST2M(ST,RETURN_VALUE(J)%MIM)
        ENDIF
    ENDDO
END FUNCTION IM_Q1

```

```

FUNCTION IM_ZQ1(C)      RESULT (RETURN_VALUE)
    USE FMVALS
    IMPLICIT NONE
    COMPLEX (QUAD_FP), DIMENSION(:) :: C
    TYPE (IM), DIMENSION(SIZE(C)) :: RETURN_VALUE
    REAL (QUAD_FP) :: D
    CHARACTER(50) :: ST
    INTEGER :: IVAL,J,N
    INTENT (IN) :: C
    N = SIZE(C)
    DO J = 1, N
        D = REAL(C(J),QUAD_FP)

```