```
 MODULE FM_INTERVAL_ARITHMETIC_1


!  FM_interval 1.4                        David M. Smith                        Interval Arithmetic

!  This module extends the definition of the basic Fortran arithmetic and function operations so
!  they also apply to multiple precision intervals, using version 1.4 of FM.
!  The multiple precision interval data type is called
!    TYPE (FM_INTERVAL)

!  Each FM interval consists of two endpoints, with each endpoint being a TYPE(FM) multiple
!  precision number.  The first of these endpoints defines the left endpoint of an interval,
!  and the second defines the right endpoint of the interval.

!  Most of the functions defined in this module are multiple precision interval versions of standard
!  Fortran functions.  In addition, there are functions for direct conversion, formatting, and some
!  mathematical special functions.

!  TO_FM_INTERVAL is a function for converting other types of numbers to type FM_INTERVAL.
!  Like the TO_FM function in module FMZM, TO_FM_INTERVAL(3.12) converts the REAL constant
!  to an FM interval, but it is accurate only to single precision.  TO_FM_INTERVAL(3.12D0)
!  agrees with 3.12 to double precision accuracy, and TO_FM_INTERVAL('3.12') or
!  TO_FM_INTERVAL(312)/TO_FM_INTERVAL(100) agrees to full FM accuracy.


    USE FMZM


!  For all comparisons except == and /=, the order is not well defined if intervals overlap.
!  In those cases, the midpoints of the intervals are compared.

    TYPE FM_INTERVAL
         TYPE(MULTI) :: LEFT
         TYPE(MULTI) :: RIGHT
    END TYPE

!            Work variables for derived type operations.

    TYPE (FM_INTERVAL), SAVE :: MTFM_I, MUFM_I, MVFM_I, MWFM_I, M0FM_I, M1FM_I, M2FM_I, M3FM_I
    TYPE (FM_INTERVAL), SAVE :: M4FM_I, M5FM_I, M6FM_I, M7FM_I, M8FM_I, M9FM_I
    TYPE (FM), SAVE :: M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_10, M_11, M_12,  &
                       X_EDGE, Y_EDGE, XY_EDGE, F_LEFT, F_RIGHT
    TYPE (ZM), SAVE :: MZ_1
    TYPE(MULTI), SAVE :: MTIM_I, MTZM_I(2)
    INTEGER, PARAMETER :: N_PREV = 10
    INTEGER, SAVE :: NDIG_XY_EDGE, KXY_EDGE, K_ROUTINE_EDGE, KROUND_PREV(0:N_PREV-1),  &
                     ROUTINE_PREV(0:N_PREV-1), NUM_PREV = 0
    TYPE (FM), SAVE :: M1_PREV(0:N_PREV-1), M2_PREV(0:N_PREV-1),  &
                       M3_PREV(0:N_PREV-1)

   INTERFACE TO_FM_INTERVAL

!            Create an interval by giving both endpoints.

      MODULE PROCEDURE INTERVAL_FM_I
      MODULE PROCEDURE INTERVAL_FM_R
      MODULE PROCEDURE INTERVAL_FM_D
```

```
      MODULE PROCEDURE INTERVAL_FM_Z
      MODULE PROCEDURE INTERVAL_FM_ZD
      MODULE PROCEDURE INTERVAL_FM_FM
      MODULE PROCEDURE INTERVAL_FM_IM
      MODULE PROCEDURE INTERVAL_FM_ZM
      MODULE PROCEDURE INTERVAL_FM_ST

!             Convert single values to intervals with both endpoints the same.

      MODULE PROCEDURE FM_INTERVAL_I
      MODULE PROCEDURE FM_INTERVAL_R
      MODULE PROCEDURE FM_INTERVAL_D
      MODULE PROCEDURE FM_INTERVAL_Z
      MODULE PROCEDURE FM_INTERVAL_ZD
      MODULE PROCEDURE FM_INTERVAL_FM
      MODULE PROCEDURE FM_INTERVAL_FMA
      MODULE PROCEDURE FM_INTERVAL_IM
      MODULE PROCEDURE FM_INTERVAL_ZM
      MODULE PROCEDURE FM_INTERVAL_ST
      MODULE PROCEDURE FM_INTERVAL_I1
      MODULE PROCEDURE FM_INTERVAL_R1
      MODULE PROCEDURE FM_INTERVAL_D1
      MODULE PROCEDURE FM_INTERVAL_Z1
      MODULE PROCEDURE FM_INTERVAL_ZD1
      MODULE PROCEDURE FM_INTERVAL_FM1
      MODULE PROCEDURE FM_INTERVAL_FMA1
      MODULE PROCEDURE FM_INTERVAL_IM1
      MODULE PROCEDURE FM_INTERVAL_ZM1
      MODULE PROCEDURE FM_INTERVAL_ST1
      MODULE PROCEDURE FM_INTERVAL_I2
      MODULE PROCEDURE FM_INTERVAL_R2
      MODULE PROCEDURE FM_INTERVAL_D2
      MODULE PROCEDURE FM_INTERVAL_Z2
      MODULE PROCEDURE FM_INTERVAL_ZD2
      MODULE PROCEDURE FM_INTERVAL_FM2
      MODULE PROCEDURE FM_INTERVAL_FMA2
      MODULE PROCEDURE FM_INTERVAL_IM2
      MODULE PROCEDURE FM_INTERVAL_ZM2
      MODULE PROCEDURE FM_INTERVAL_ST2
   END INTERFACE

!             Return the left or right endpoint of an interval as a type (fm) number.

   INTERFACE LEFT_ENDPOINT
      MODULE PROCEDURE LEFT_ENDPOINT_INTERVAL_FM
   END INTERFACE

   INTERFACE RIGHT_ENDPOINT
      MODULE PROCEDURE RIGHT_ENDPOINT_INTERVAL_FM
   END INTERFACE

   INTERFACE TO_FM
      MODULE PROCEDURE FM_FM_INTERVAL
      MODULE PROCEDURE FM_FM_INTERVAL1
      MODULE PROCEDURE FM_FM_INTERVAL2
   END INTERFACE

   INTERFACE TO_IM
```

```
   MODULE PROCEDURE IM_FM_INTERVAL
   MODULE PROCEDURE IM_FM_INTERVAL1
   MODULE PROCEDURE IM_FM_INTERVAL2
END INTERFACE

INTERFACE TO_ZM
   MODULE PROCEDURE ZM_FM_INTERVAL
   MODULE PROCEDURE ZM_FM_INTERVAL1
   MODULE PROCEDURE ZM_FM_INTERVAL2
END INTERFACE

INTERFACE TO_INT
   MODULE PROCEDURE FM_INTERVAL_2INT
   MODULE PROCEDURE FM_INTERVAL_2INT1
   MODULE PROCEDURE FM_INTERVAL_2INT2
END INTERFACE

INTERFACE TO_SP
   MODULE PROCEDURE FM_INTERVAL_2SP
   MODULE PROCEDURE FM_INTERVAL_2SP1
   MODULE PROCEDURE FM_INTERVAL_2SP2
END INTERFACE

INTERFACE TO_DP
   MODULE PROCEDURE FM_INTERVAL_2DP
   MODULE PROCEDURE FM_INTERVAL_2DP1
   MODULE PROCEDURE FM_INTERVAL_2DP2
END INTERFACE

INTERFACE TO_SPZ
   MODULE PROCEDURE FM_INTERVAL_2SPZ
   MODULE PROCEDURE FM_INTERVAL_2SPZ1
   MODULE PROCEDURE FM_INTERVAL_2SPZ2
END INTERFACE

INTERFACE TO_DPZ
   MODULE PROCEDURE FM_INTERVAL_2DPZ
   MODULE PROCEDURE FM_INTERVAL_2DPZ1
   MODULE PROCEDURE FM_INTERVAL_2DPZ2
END INTERFACE

INTERFACE IS_OVERFLOW
   MODULE PROCEDURE FM_INTERVAL_IS_OVERFLOW
   MODULE PROCEDURE FM_INTERVAL_IS_OVERFLOW1
   MODULE PROCEDURE FM_INTERVAL_IS_OVERFLOW2
END INTERFACE

INTERFACE IS_UNDERFLOW
   MODULE PROCEDURE FM_INTERVAL_IS_UNDERFLOW
   MODULE PROCEDURE FM_INTERVAL_IS_UNDERFLOW1
   MODULE PROCEDURE FM_INTERVAL_IS_UNDERFLOW2
END INTERFACE

INTERFACE IS_UNKNOWN
   MODULE PROCEDURE FM_INTERVAL_IS_UNKNOWN
   MODULE PROCEDURE FM_INTERVAL_IS_UNKNOWN1
   MODULE PROCEDURE FM_INTERVAL_IS_UNKNOWN2
END INTERFACE
```

```
  INTERFACE FM_INTERVAL_UNDEF_INP
     MODULE PROCEDURE FM_UNDEF_INP_INTERVAL_FM0
     MODULE PROCEDURE FM_UNDEF_INP_INTERVAL_FM1
     MODULE PROCEDURE FM_UNDEF_INP_INTERVAL_FM2
  END INTERFACE

 CONTAINS

!                                                       TO_FM_INTERVAL

  FUNCTION FM_INTERVAL_I(IVAL)     RESULT (RETURN_VALUE)
     USE FMVALS
     IMPLICIT NONE
     TYPE (FM_INTERVAL) :: RETURN_VALUE
     INTEGER :: IVAL
     INTENT (IN) :: IVAL
     CALL FMI2M_INTERVAL(IVAL,RETURN_VALUE)
  END FUNCTION FM_INTERVAL_I

  FUNCTION FM_INTERVAL_R(R)     RESULT (RETURN_VALUE)
     USE FMVALS
     IMPLICIT NONE
     TYPE (FM_INTERVAL) :: RETURN_VALUE
     REAL :: R
     INTENT (IN) :: R
     CALL FMSP2M_INTERVAL(R,RETURN_VALUE)
  END FUNCTION FM_INTERVAL_R

  FUNCTION FM_INTERVAL_D(D)     RESULT (RETURN_VALUE)
     USE FMVALS
     IMPLICIT NONE
     TYPE (FM_INTERVAL) :: RETURN_VALUE
     DOUBLE PRECISION :: D
     INTENT (IN) :: D
     CALL FMDP2M_INTERVAL(D,RETURN_VALUE)
  END FUNCTION FM_INTERVAL_D

  FUNCTION FM_INTERVAL_Z(Z)     RESULT (RETURN_VALUE)
     USE FMVALS
     IMPLICIT NONE
     TYPE (FM_INTERVAL) :: RETURN_VALUE
     COMPLEX :: Z
     INTENT (IN) :: Z
     CALL FMSP2M_INTERVAL(REAL(Z),RETURN_VALUE)
  END FUNCTION FM_INTERVAL_Z

  FUNCTION FM_INTERVAL_ZD(C)     RESULT (RETURN_VALUE)
     USE FMVALS
     IMPLICIT NONE
     TYPE (FM_INTERVAL) :: RETURN_VALUE
     COMPLEX (KIND(0.0D0)) :: C
     INTENT (IN) :: C
     CALL FMDP2M_INTERVAL(REAL(C,KIND(0.0D0)),RETURN_VALUE)
  END FUNCTION FM_INTERVAL_ZD

  FUNCTION FM_INTERVAL_FM(MA)     RESULT (RETURN_VALUE)
     USE FMVALS
```

```
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: MA,RETURN_VALUE
   INTENT (IN) :: MA
   CALL FM_INTERVAL_UNDEF_INP(MA)
   CALL FMMIN(MA%LEFT,MA%RIGHT,RETURN_VALUE%LEFT)
   CALL FMMAX(MA%LEFT,MA%RIGHT,RETURN_VALUE%RIGHT)
END FUNCTION FM_INTERVAL_FM

FUNCTION FM_INTERVAL_FMA(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   TYPE (FM) :: MA
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   CALL FMEQ(MA%MFM,RETURN_VALUE%LEFT)
   CALL FMEQ(MA%MFM,RETURN_VALUE%RIGHT)
END FUNCTION FM_INTERVAL_FMA

FUNCTION FM_INTERVAL_IM(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   TYPE (IM) :: MA
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   CALL IMI2FM(MA%MIM,RETURN_VALUE%LEFT)
   CALL IMI2FM(MA%MIM,RETURN_VALUE%RIGHT)
END FUNCTION FM_INTERVAL_IM

FUNCTION FM_INTERVAL_ZM(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   TYPE (ZM) :: MA
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   CALL ZMREAL_INTERVAL(MA%MZM,RETURN_VALUE)
END FUNCTION FM_INTERVAL_ZM

FUNCTION FM_INTERVAL_ST(ST)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   CHARACTER(*) :: ST
   INTENT (IN) :: ST
   CALL FMST2M_INTERVAL(ST,RETURN_VALUE)
END FUNCTION FM_INTERVAL_ST

FUNCTION FM_INTERVAL_I1(IVAL)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   INTEGER, DIMENSION(:) :: IVAL
   TYPE (FM_INTERVAL), DIMENSION(SIZE(IVAL)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: IVAL
   N = SIZE(IVAL)
   DO J = 1, N
```

```
      CALL FMI2M_INTERVAL(IVAL(J),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_I1

FUNCTION FM_INTERVAL_R1(R)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   REAL, DIMENSION(:) :: R
   TYPE (FM_INTERVAL), DIMENSION(SIZE(R)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: R
   N = SIZE(R)
   DO J = 1, N
      CALL FMSP2M_INTERVAL(R(J),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_R1

FUNCTION FM_INTERVAL_D1(D)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   DOUBLE PRECISION, DIMENSION(:) :: D
   TYPE (FM_INTERVAL), DIMENSION(SIZE(D)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: D
   N = SIZE(D)
   DO J = 1, N
      CALL FMDP2M_INTERVAL(D(J),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_D1

FUNCTION FM_INTERVAL_Z1(Z)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX, DIMENSION(:) :: Z
   TYPE (FM_INTERVAL), DIMENSION(SIZE(Z)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: Z
   N = SIZE(Z)
   DO J = 1, N
      CALL FMSP2M_INTERVAL(REAL(Z(J)),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_Z1

FUNCTION FM_INTERVAL_ZD1(C)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX (KIND(0.0D0)), DIMENSION(:) :: C
   TYPE (FM_INTERVAL), DIMENSION(SIZE(C)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: C
   N = SIZE(C)
   DO J = 1, N
      CALL FMDP2M_INTERVAL(REAL(C(J),KIND(0.0D0)),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_ZD1

FUNCTION FM_INTERVAL_FM1(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
```

```
   IMPLICIT NONE
   TYPE (FM_INTERVAL), DIMENSION(:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: MA
   CALL FM_INTERVAL_UNDEF_INP(MA)
   N = SIZE(MA)
   DO J = 1, N
      CALL FMEQ_INTERVAL(MA(J),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_FM1

FUNCTION FM_INTERVAL_FMA1(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM), DIMENSION(:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   N = SIZE(MA)
   DO J = 1, N
      CALL FMEQ(MA(J)%MFM,RETURN_VALUE(J)%LEFT)
      CALL FMEQ(MA(J)%MFM,RETURN_VALUE(J)%RIGHT)
   ENDDO
END FUNCTION FM_INTERVAL_FMA1

FUNCTION FM_INTERVAL_IM1(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (IM), DIMENSION(:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   N = SIZE(MA)
   DO J = 1, N
      CALL IMI2FM_INTERVAL(MA(J)%MIM,RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_IM1

FUNCTION FM_INTERVAL_ZM1(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (ZM), DIMENSION(:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   N = SIZE(MA)
   DO J = 1, N
      CALL ZMREAL_INTERVAL(MA(J)%MZM,RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_ZM1

FUNCTION FM_INTERVAL_ST1(ST)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
```

```
   CHARACTER(*), DIMENSION(:) :: ST
   TYPE (FM_INTERVAL), DIMENSION(SIZE(ST)) :: RETURN_VALUE
   INTEGER :: J,N
   INTENT (IN) :: ST
   N = SIZE(ST)
   DO J = 1, N
      CALL FMST2M_INTERVAL(ST(J),RETURN_VALUE(J))
   ENDDO
END FUNCTION FM_INTERVAL_ST1

FUNCTION FM_INTERVAL_I2(IVAL)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   INTEGER, DIMENSION(:,:) :: IVAL
   TYPE (FM_INTERVAL), DIMENSION(SIZE(IVAL,DIM=1),SIZE(IVAL,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: IVAL
   DO J = 1, SIZE(IVAL,DIM=1)
      DO K = 1, SIZE(IVAL,DIM=2)
         CALL FMI2M_INTERVAL(IVAL(J,K),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_I2

FUNCTION FM_INTERVAL_R2(R)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   REAL, DIMENSION(:,:) :: R
   TYPE (FM_INTERVAL), DIMENSION(SIZE(R,DIM=1),SIZE(R,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: R
   DO J = 1, SIZE(R,DIM=1)
      DO K = 1, SIZE(R,DIM=2)
         CALL FMSP2M_INTERVAL(R(J,K),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_R2

FUNCTION FM_INTERVAL_D2(D)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   DOUBLE PRECISION, DIMENSION(:,:) :: D
   TYPE (FM_INTERVAL), 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 FMDP2M_INTERVAL(D(J,K),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_D2

FUNCTION FM_INTERVAL_Z2(Z)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX, DIMENSION(:,:) :: Z
   TYPE (FM_INTERVAL), DIMENSION(SIZE(Z,DIM=1),SIZE(Z,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
```

```
   INTENT (IN) :: Z
   DO J = 1, SIZE(Z,DIM=1)
      DO K = 1, SIZE(Z,DIM=2)
         CALL FMSP2M_INTERVAL(REAL(Z(J,K)),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_Z2

FUNCTION FM_INTERVAL_ZD2(C)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX (KIND(0.0D0)), DIMENSION(:,:) :: C
   TYPE (FM_INTERVAL), 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 FMDP2M_INTERVAL(REAL(C(J,K),KIND(0.0D0)),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_ZD2

FUNCTION FM_INTERVAL_FM2(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: MA
   CALL FM_INTERVAL_UNDEF_INP(MA)
   DO J = 1, SIZE(MA,DIM=1)
      DO K = 1, SIZE(MA,DIM=2)
         CALL FMEQ_INTERVAL(MA(J,K),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_FM2

FUNCTION FM_INTERVAL_FMA2(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   DO J = 1, SIZE(MA,DIM=1)
      DO K = 1, SIZE(MA,DIM=2)
         CALL FMEQ(MA(J,K)%MFM,RETURN_VALUE(J,K)%LEFT)
         CALL FMEQ(MA(J,K)%MFM,RETURN_VALUE(J,K)%RIGHT)
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_FMA2

FUNCTION FM_INTERVAL_IM2(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (IM), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
```

```
   INTEGER :: J,K
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   DO J = 1, SIZE(MA,DIM=1)
      DO K = 1, SIZE(MA,DIM=2)
         CALL IMI2FM_INTERVAL(MA(J,K)%MIM,RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_IM2

FUNCTION FM_INTERVAL_ZM2(MA)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (ZM), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: MA
   CALL FM_UNDEF_INP(MA)
   DO J = 1, SIZE(MA,DIM=1)
      DO K = 1, SIZE(MA,DIM=2)
         CALL ZMREAL_INTERVAL(MA(J,K)%MZM,RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_ZM2

FUNCTION FM_INTERVAL_ST2(ST)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   CHARACTER(*), DIMENSION(:,:) :: ST
   TYPE (FM_INTERVAL), DIMENSION(SIZE(ST,DIM=1),SIZE(ST,DIM=2)) :: RETURN_VALUE
   INTEGER :: J,K
   INTENT (IN) :: ST
   DO J = 1, SIZE(ST,DIM=1)
      DO K = 1, SIZE(ST,DIM=2)
         CALL FMST2M_INTERVAL(ST(J,K),RETURN_VALUE(J,K))
      ENDDO
   ENDDO
END FUNCTION FM_INTERVAL_ST2

FUNCTION INTERVAL_FM_I(IVAL1,IVAL2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   INTEGER :: IVAL1,IVAL2,IV1,IV2
   INTENT (IN) :: IVAL1,IVAL2
   IV1 = MIN(IVAL1,IVAL2)
   IV2 = MAX(IVAL1,IVAL2)
   CALL FMI2M(IV1,RETURN_VALUE%LEFT)
   CALL FMI2M(IV2,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_I

FUNCTION INTERVAL_FM_R(R1,R2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   REAL :: R1,R2,RV1,RV2
   INTENT (IN) :: R1,R2
   RV1 = MIN(R1,R2)
```

```
   RV2 = MAX(R1,R2)
   CALL FMSP2M(RV1,RETURN_VALUE%LEFT)
   CALL FMSP2M(RV2,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_R

FUNCTION INTERVAL_FM_D(D1,D2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   DOUBLE PRECISION :: D1,D2,DV1,DV2
   INTENT (IN) :: D1,D2
   DV1 = MIN(D1,D2)
   DV2 = MAX(D1,D2)
   CALL FMDP2M(DV1,RETURN_VALUE%LEFT)
   CALL FMDP2M(DV2,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_D

FUNCTION INTERVAL_FM_Z(Z1,Z2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   COMPLEX :: Z1,Z2
   REAL :: RV1,RV2
   INTENT (IN) :: Z1,Z2
   RV1 = MIN(REAL(Z1),REAL(Z2))
   RV2 = MAX(REAL(Z1),REAL(Z2))
   CALL FMSP2M(RV1,RETURN_VALUE%LEFT)
   CALL FMSP2M(RV2,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_Z

FUNCTION INTERVAL_FM_ZD(C1,C2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   COMPLEX (KIND(0.0D0)) :: C1,C2
   DOUBLE PRECISION :: DV1,DV2
   INTENT (IN) :: C1,C2
   DV1 = MIN(REAL(C1,KIND(0.0D0)),REAL(C2,KIND(0.0D0)))
   DV2 = MAX(REAL(C1,KIND(0.0D0)),REAL(C2,KIND(0.0D0)))
   CALL FMDP2M(DV1,RETURN_VALUE%LEFT)
   CALL FMDP2M(DV2,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_ZD

FUNCTION INTERVAL_FM_FM(M1,M2)     RESULT (RETURN_VALUE)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: RETURN_VALUE
   TYPE (FM) :: M1,M2
   INTENT (IN) :: M1,M2
   TYPE(MULTI), SAVE :: MTLVFM,MULVFM
   CALL FM_UNDEF_INP(M1)
   CALL FM_UNDEF_INP(M2)
   CALL FMMIN(M1%MFM,M2%MFM,MTLVFM)
   CALL FMMAX(M1%MFM,M2%MFM,MULVFM)
   CALL FMEQ(MTLVFM,RETURN_VALUE%LEFT)
   CALL FMEQ(MULVFM,RETURN_VALUE%RIGHT)
END FUNCTION INTERVAL_FM_FM
```

```
   FUNCTION INTERVAL_FM_IM(M1,M2)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: RETURN_VALUE
      TYPE (IM) :: M1,M2
      INTENT (IN) :: M1,M2
      TYPE(MULTI), SAVE :: MTLVIM,MULVIM
      CALL FM_UNDEF_INP(M1)
      CALL FM_UNDEF_INP(M2)
      CALL IMMIN(M1%MIM,M2%MIM,MTLVIM)
      CALL IMMAX(M1%MIM,M2%MIM,MULVIM)
      CALL IMI2FM(MTLVIM,RETURN_VALUE%LEFT)
      CALL IMI2FM(MULVIM,RETURN_VALUE%RIGHT)
   END FUNCTION INTERVAL_FM_IM

   FUNCTION INTERVAL_FM_ZM(M1,M2)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: RETURN_VALUE
      TYPE (ZM) :: M1,M2
      INTENT (IN) :: M1,M2
      TYPE(MULTI), SAVE :: M1LVFM,M2LVFM
      CALL FM_UNDEF_INP(M1)
      CALL FM_UNDEF_INP(M2)
      CALL ZMREAL(M1%MZM,M1LVFM)
      CALL ZMREAL(M2%MZM,M2LVFM)
      CALL FMMIN(M1LVFM,M2LVFM,RETURN_VALUE%LEFT)
      CALL FMMAX(M1LVFM,M2LVFM,RETURN_VALUE%RIGHT)
   END FUNCTION INTERVAL_FM_ZM

   FUNCTION INTERVAL_FM_ST(S1,S2)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: RETURN_VALUE
      CHARACTER(*) :: S1,S2
      INTENT (IN) :: S1,S2
      TYPE(MULTI), SAVE :: M1LVFM,M2LVFM
      CALL FMST2M(S1,M1LVFM)
      CALL FMST2M(S2,M2LVFM)
      CALL FMMIN(M1LVFM,M2LVFM,RETURN_VALUE%LEFT)
      CALL FMMAX(M1LVFM,M2LVFM,RETURN_VALUE%RIGHT)
   END FUNCTION INTERVAL_FM_ST

!                                                                LEFT_ENDPOINT

   FUNCTION LEFT_ENDPOINT_INTERVAL_FM(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FMEQ(MA%LEFT,RETURN_VALUE%MFM)
   END FUNCTION LEFT_ENDPOINT_INTERVAL_FM

!                                                               RIGHT_ENDPOINT

   FUNCTION RIGHT_ENDPOINT_INTERVAL_FM(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
```

```
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FMEQ(MA%RIGHT,RETURN_VALUE%MFM)
   END FUNCTION RIGHT_ENDPOINT_INTERVAL_FM

!                                                                  TO_FM

   FUNCTION FM_FM_INTERVAL(MA)     RESULT (RETURN_VALUE)

!  When converting an interval to a non-interval value, use the midpoint.

      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: RETURN_VALUE
      INTENT (IN) :: MA
      TYPE(MULTI), SAVE :: MTLVFM
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMSUB(MA%RIGHT,MA%LEFT,MTLVFM)
      CALL FMDIVI_R1(MTLVFM,2)
      CALL FMADD(MA%LEFT,MTLVFM,RETURN_VALUE%MFM)
   END FUNCTION FM_FM_INTERVAL

   FUNCTION FM_FM_INTERVAL1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      TYPE (FM), DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      TYPE(MULTI), SAVE :: MTLVFM
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMSUB(MA(J)%RIGHT,MA(J)%LEFT,MTLVFM)
         CALL FMDIVI_R1(MTLVFM,2)
         CALL FMADD(MA(J)%LEFT,MTLVFM,RETURN_VALUE(J)%MFM)
      ENDDO
   END FUNCTION FM_FM_INTERVAL1

   FUNCTION FM_FM_INTERVAL2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (FM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      TYPE(MULTI), SAVE :: MTLVFM
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMSUB(MA(J,K)%RIGHT,MA(J,K)%LEFT,MTLVFM)
            CALL FMDIVI_R1(MTLVFM,2)
            CALL FMADD(MA(J,K)%LEFT,MTLVFM,RETURN_VALUE(J,K)%MFM)
         ENDDO
      ENDDO
```

```
   END FUNCTION FM_FM_INTERVAL2

!                                                                  TO_IM

   FUNCTION IM_FM_INTERVAL(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (IM) :: RETURN_VALUE
      TYPE (FM_INTERVAL) :: MA
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL IMFM2I_INTERVAL(MA,RETURN_VALUE%MIM)
   END FUNCTION IM_FM_INTERVAL

   FUNCTION IM_FM_INTERVAL1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      TYPE (IM), DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL IMFM2I_INTERVAL(MA(J),RETURN_VALUE(J)%MIM)
      ENDDO
   END FUNCTION IM_FM_INTERVAL1

   FUNCTION IM_FM_INTERVAL2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (IM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL IMFM2I_INTERVAL(MA(J,K),RETURN_VALUE(J,K)%MIM)
         ENDDO
      ENDDO
   END FUNCTION IM_FM_INTERVAL2

!                                                                  TO_ZM

   FUNCTION ZM_FM_INTERVAL(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (ZM) :: RETURN_VALUE
      TYPE (FM_INTERVAL) :: MA
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMI2M_INTERVAL(0,MUFM_I)
      CALL ZMCMPX_INTERVAL(MA,MUFM_I,RETURN_VALUE)
   END FUNCTION ZM_FM_INTERVAL

   FUNCTION ZM_FM_INTERVAL1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
```

```
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      TYPE (ZM), DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      CALL FMI2M_INTERVAL(0,MUFM_I)
      DO J = 1, N
         CALL ZMCMPX_INTERVAL(MA(J),MUFM_I,RETURN_VALUE(J))
      ENDDO
   END FUNCTION ZM_FM_INTERVAL1

   FUNCTION ZM_FM_INTERVAL2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (ZM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMI2M_INTERVAL(0,MUFM_I)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL ZMCMPX_INTERVAL(MA(J,K),MUFM_I,RETURN_VALUE(J,K))
         ENDDO
      ENDDO
   END FUNCTION ZM_FM_INTERVAL2

!                                                                  TO_INT

   FUNCTION FM_INTERVAL_2INT(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      INTEGER :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2I_INTERVAL(MA,RETURN_VALUE)
   END FUNCTION FM_INTERVAL_2INT

   FUNCTION FM_INTERVAL_2INT1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      INTEGER, DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMM2I_INTERVAL(MA(J),RETURN_VALUE(J))
      ENDDO
   END FUNCTION FM_INTERVAL_2INT1

   FUNCTION FM_INTERVAL_2INT2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
```

```
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      INTEGER, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMM2I_INTERVAL(MA(J,K),RETURN_VALUE(J,K))
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_2INT2

!                                                                  TO_SP

   FUNCTION FM_INTERVAL_2SP(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      REAL :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2SP_INTERVAL(MA,RETURN_VALUE)
   END FUNCTION FM_INTERVAL_2SP

   FUNCTION FM_INTERVAL_2SP1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      REAL, DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMM2SP_INTERVAL(MA(J),RETURN_VALUE(J))
      ENDDO
   END FUNCTION FM_INTERVAL_2SP1

   FUNCTION FM_INTERVAL_2SP2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      REAL, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMM2SP_INTERVAL(MA(J,K),RETURN_VALUE(J,K))
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_2SP2

!                                                                  TO_DP

   FUNCTION FM_INTERVAL_2DP(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
```

```
      TYPE (FM_INTERVAL) :: MA
      DOUBLE PRECISION :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2DP_INTERVAL(MA,RETURN_VALUE)
   END FUNCTION FM_INTERVAL_2DP

   FUNCTION FM_INTERVAL_2DP1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      DOUBLE PRECISION, DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMM2DP_INTERVAL(MA(J),RETURN_VALUE(J))
      ENDDO
   END FUNCTION FM_INTERVAL_2DP1

   FUNCTION FM_INTERVAL_2DP2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      DOUBLE PRECISION, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMM2DP_INTERVAL(MA(J,K),RETURN_VALUE(J,K))
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_2DP2

!                                                                  TO_SPZ

   FUNCTION FM_INTERVAL_2SPZ(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      COMPLEX :: RETURN_VALUE
      REAL :: R
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2SP_INTERVAL(MA,R)
      RETURN_VALUE = CMPLX( R , 0.0 )
   END FUNCTION FM_INTERVAL_2SPZ

   FUNCTION FM_INTERVAL_2SPZ1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      COMPLEX, DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      REAL :: R
      INTENT (IN) :: MA
```

```
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMM2SP_INTERVAL(MA(J),R)
         RETURN_VALUE(J) = CMPLX( R , 0.0 )
      ENDDO
   END FUNCTION FM_INTERVAL_2SPZ1

   FUNCTION FM_INTERVAL_2SPZ2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      COMPLEX, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      REAL :: R
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMM2SP_INTERVAL(MA(J,K),R)
            RETURN_VALUE(J,K) = CMPLX( R , 0.0 )
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_2SPZ2

!                                                                  TO_DPZ

   FUNCTION FM_INTERVAL_2DPZ(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      COMPLEX (KIND(0.0D0)) :: RETURN_VALUE
      DOUBLE PRECISION :: D
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2DP_INTERVAL(MA,D)
      RETURN_VALUE = CMPLX( D , 0.0D0 , KIND(0.0D0) )
   END FUNCTION FM_INTERVAL_2DPZ

   FUNCTION FM_INTERVAL_2DPZ1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      COMPLEX (KIND(0.0D0)), DIMENSION(SIZE(MA)) :: RETURN_VALUE
      INTEGER :: J,N
      DOUBLE PRECISION :: D
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      DO J = 1, N
         CALL FMM2DP_INTERVAL(MA(J),D)
         RETURN_VALUE(J) = CMPLX( D , 0.0D0 , KIND(0.0D0) )
      ENDDO
   END FUNCTION FM_INTERVAL_2DPZ1

   FUNCTION FM_INTERVAL_2DPZ2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
```

```
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      COMPLEX (KIND(0.0D0)), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: RETURN_VALUE
      INTEGER :: J,K
      DOUBLE PRECISION :: D
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            CALL FMM2DP_INTERVAL(MA(J,K),D)
            RETURN_VALUE(J,K) = CMPLX( D , 0.0D0 , KIND(0.0D0) )
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_2DPZ2

!                                                          IS_OVERFLOW

   FUNCTION FM_INTERVAL_IS_OVERFLOW(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      LOGICAL :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      RETURN_VALUE = .FALSE.
      IF (MA%LEFT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
      IF (MA%RIGHT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
   END FUNCTION FM_INTERVAL_IS_OVERFLOW

   FUNCTION FM_INTERVAL_IS_OVERFLOW1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      LOGICAL :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      RETURN_VALUE = .FALSE.
      DO J = 1, N
         IF (MA(J)%LEFT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
         IF (MA(J)%RIGHT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
      ENDDO
   END FUNCTION FM_INTERVAL_IS_OVERFLOW1

   FUNCTION FM_INTERVAL_IS_OVERFLOW2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      LOGICAL :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      RETURN_VALUE = .FALSE.
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            IF (MA(J,K)%LEFT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
            IF (MA(J,K)%RIGHT%MP(2) == MEXPOV) RETURN_VALUE = .TRUE.
         ENDDO
```

```
      ENDDO
   END FUNCTION FM_INTERVAL_IS_OVERFLOW2

!                                                          IS_UNDERFLOW

   FUNCTION FM_INTERVAL_IS_UNDERFLOW(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      LOGICAL :: RETURN_VALUE
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      RETURN_VALUE = .FALSE.
      IF (MA%LEFT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
      IF (MA%RIGHT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
   END FUNCTION FM_INTERVAL_IS_UNDERFLOW

!  The integer versions are included for completeness, but type (im) numbers can't underflow.

   FUNCTION FM_INTERVAL_IS_UNDERFLOW1(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      LOGICAL :: RETURN_VALUE
      INTEGER :: J,N
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      N = SIZE(MA)
      RETURN_VALUE = .FALSE.
      DO J = 1, N
         IF (MA(J)%LEFT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
         IF (MA(J)%RIGHT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
      ENDDO
   END FUNCTION FM_INTERVAL_IS_UNDERFLOW1

   FUNCTION FM_INTERVAL_IS_UNDERFLOW2(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      LOGICAL :: RETURN_VALUE
      INTEGER :: J,K
      INTENT (IN) :: MA
      CALL FM_INTERVAL_UNDEF_INP(MA)
      RETURN_VALUE = .FALSE.
      DO J = 1, SIZE(MA,DIM=1)
         DO K = 1, SIZE(MA,DIM=2)
            IF (MA(J,K)%LEFT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
            IF (MA(J,K)%RIGHT%MP(2) == MEXPUN) RETURN_VALUE = .TRUE.
         ENDDO
      ENDDO
   END FUNCTION FM_INTERVAL_IS_UNDERFLOW2

!                                                          IS_UNKNOWN

   FUNCTION FM_INTERVAL_IS_UNKNOWN(MA)     RESULT (RETURN_VALUE)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
```