```
 MODULE FM_INTERVAL_ARITHMETIC_1


!  FM_interval 1.3                        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 1.3 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
         INTEGER :: LEFT = -1
         INTEGER :: RIGHT = -1
    END TYPE

!             Work variables for derived type operations.

    TYPE (FM_INTERVAL), SAVE :: MTFM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: MUFM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: MVFM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: MWFM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M0FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M1FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M2FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M3FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M4FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M5FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M6FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M7FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M8FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM_INTERVAL), SAVE :: M9FM_I = FM_INTERVAL(-3,-3)
    TYPE (FM), SAVE :: M_1 = FM(-3), M_2 = FM(-3), M_3 = FM(-3), M_4 = FM(-3), M_5 = FM(-3),   &
                       M_6 = FM(-3), M_7 = FM(-3), M_8 = FM(-3), M_9 = FM(-3), M_10 = FM(-3),  &
                       M_11 = FM(-3), M_12 = FM(-3), X_EDGE = FM(-3), Y_EDGE = FM(-3),         &
                       XY_EDGE = FM(-3), F_LEFT = FM(-3), F_RIGHT = FM(-3)
    TYPE (ZM), SAVE :: MZ_1 = ZM( (/ -3, -3 /) )
    INTEGER, SAVE :: MTIM_I = -3
    INTEGER, SAVE :: MTZM_I(2) = (/ -3, -3 /)
    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) = FM(-3), M2_PREV(0:N_PREV-1) = FM(-3),  &
                       M3_PREV(0:N_PREV-1) = FM(-3)

   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
```

```fortran
   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 FMEQ_INDEX_INTERVAL
      MODULE PROCEDURE FMEQ_INDEX_INTERVAL_FM0
      MODULE PROCEDURE FMEQ_INDEX_INTERVAL_FM1
      MODULE PROCEDURE FMEQ_INDEX_INTERVAL_FM2
   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)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_I
      INTEGER :: IVAL
      INTENT (IN) :: IVAL
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMI2M_INTERVAL(IVAL,FM_INTERVAL_I)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_I

   FUNCTION FM_INTERVAL_R(R)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_R
      REAL :: R
      INTENT (IN) :: R
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMSP2M_INTERVAL(R,FM_INTERVAL_R)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_R

   FUNCTION FM_INTERVAL_D(D)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_D
      DOUBLE PRECISION :: D
      INTENT (IN) :: D
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMDP2M_INTERVAL(D,FM_INTERVAL_D)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_D

   FUNCTION FM_INTERVAL_Z(Z)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_Z
      COMPLEX :: Z
      INTENT (IN) :: Z
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMSP2M_INTERVAL(REAL(Z),FM_INTERVAL_Z)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_Z

   FUNCTION FM_INTERVAL_ZD(C)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_ZD
      COMPLEX (KIND(0.0D0)) :: C
      INTENT (IN) :: C
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMDP2M_INTERVAL(REAL(C,KIND(0.0D0)),FM_INTERVAL_ZD)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_ZD

   FUNCTION FM_INTERVAL_FM(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA,FM_INTERVAL_FM
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMMIN(MA%LEFT,MA%RIGHT,FM_INTERVAL_FM%LEFT)
      CALL FMMAX(MA%LEFT,MA%RIGHT,FM_INTERVAL_FM%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_FM

   FUNCTION FM_INTERVAL_FMA(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_FMA
      TYPE (FM) :: MA
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_UNDEF_INP(MA)
      CALL FMEQ(MA%MFM,FM_INTERVAL_FMA%LEFT)
      CALL FMEQ(MA%MFM,FM_INTERVAL_FMA%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_FMA

   FUNCTION FM_INTERVAL_IM(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_IM
      TYPE (IM) :: MA
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_UNDEF_INP(MA)
      CALL IMI2FM(MA%MIM,FM_INTERVAL_IM%LEFT)
      CALL IMI2FM(MA%MIM,FM_INTERVAL_IM%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_IM

   FUNCTION FM_INTERVAL_ZM(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_ZM
      TYPE (ZM) :: MA
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_UNDEF_INP(MA)
      CALL ZMREAL_INTERVAL(MA%MZM,FM_INTERVAL_ZM)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_ZM

   FUNCTION FM_INTERVAL_ST(ST)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: FM_INTERVAL_ST
      CHARACTER(*) :: ST
      INTENT (IN) :: ST
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMST2M_INTERVAL(ST,FM_INTERVAL_ST)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_ST

   FUNCTION FM_INTERVAL_I1(IVAL)
      USE FMVALS
      IMPLICIT NONE
      INTEGER, DIMENSION(:) :: IVAL
      TYPE (FM_INTERVAL), DIMENSION(SIZE(IVAL)) :: FM_INTERVAL_I1
      INTEGER :: J,N
      INTENT (IN) :: IVAL
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      N = SIZE(IVAL)
      DO J = 1, N
         CALL FMI2M_INTERVAL(IVAL(J),FM_INTERVAL_I1(J))
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_I1
```

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

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

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

FUNCTION FM_INTERVAL_ZD1(C)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX (KIND(0.0D0)), DIMENSION(:) :: C
   TYPE (FM_INTERVAL), DIMENSION(SIZE(C)) :: FM_INTERVAL_ZD1
   INTEGER :: J,N
   INTENT (IN) :: C
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   N = SIZE(C)
   DO J = 1, N
```

```
      CALL FMDP2M_INTERVAL(REAL(C(J),KIND(0.0D0)),FM_INTERVAL_ZD1(J))
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_ZD1

FUNCTION FM_INTERVAL_FM1(MA)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL), DIMENSION(:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA)) :: FM_INTERVAL_FM1
   INTEGER :: J,N
   INTENT (IN) :: MA
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   CALL FM_INTERVAL_UNDEF_INP(MA)
   N = SIZE(MA)
   DO J = 1, N
      CALL FMEQ_INTERVAL(MA(J),FM_INTERVAL_FM1(J))
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_FM1

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

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

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

FUNCTION FM_INTERVAL_ST1(ST)
   USE FMVALS
   IMPLICIT NONE
   CHARACTER(*), DIMENSION(:) :: ST
   TYPE (FM_INTERVAL), DIMENSION(SIZE(ST)) :: FM_INTERVAL_ST1
   INTEGER :: J,N
   INTENT (IN) :: ST
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   N = SIZE(ST)
   DO J = 1, N
      CALL FMST2M_INTERVAL(ST(J),FM_INTERVAL_ST1(J))
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_ST1

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

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

FUNCTION FM_INTERVAL_D2(D)
   USE FMVALS
   IMPLICIT NONE
   DOUBLE PRECISION, DIMENSION(:,:) :: D
   TYPE (FM_INTERVAL), DIMENSION(SIZE(D,DIM=1),SIZE(D,DIM=2)) :: FM_INTERVAL_D2
   INTEGER :: J,K
   INTENT (IN) :: D
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   DO J = 1, SIZE(D,DIM=1)
      DO K = 1, SIZE(D,DIM=2)
         CALL FMDP2M_INTERVAL(D(J,K),FM_INTERVAL_D2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_D2

FUNCTION FM_INTERVAL_Z2(Z)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX, DIMENSION(:,:) :: Z
   TYPE (FM_INTERVAL), DIMENSION(SIZE(Z,DIM=1),SIZE(Z,DIM=2)) :: FM_INTERVAL_Z2
   INTEGER :: J,K
   INTENT (IN) :: Z
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   DO J = 1, SIZE(Z,DIM=1)
      DO K = 1, SIZE(Z,DIM=2)
         CALL FMSP2M_INTERVAL(REAL(Z(J,K)),FM_INTERVAL_Z2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_Z2

FUNCTION FM_INTERVAL_ZD2(C)
   USE FMVALS
   IMPLICIT NONE
   COMPLEX (KIND(0.0D0)), DIMENSION(:,:) :: C
   TYPE (FM_INTERVAL), DIMENSION(SIZE(C,DIM=1),SIZE(C,DIM=2)) :: FM_INTERVAL_ZD2
   INTEGER :: J,K
   INTENT (IN) :: C
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   DO J = 1, SIZE(C,DIM=1)
      DO K = 1, SIZE(C,DIM=2)
         CALL FMDP2M_INTERVAL(REAL(C(J,K),KIND(0.0D0)),FM_INTERVAL_ZD2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_ZD2

FUNCTION FM_INTERVAL_FM2(MA)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_FM2
   INTEGER :: J,K
   INTENT (IN) :: MA
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   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),FM_INTERVAL_FM2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_FM2

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

FUNCTION FM_INTERVAL_IM2(MA)
   USE FMVALS
   IMPLICIT NONE
   TYPE (IM), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_IM2
   INTEGER :: J,K
   INTENT (IN) :: MA
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   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,FM_INTERVAL_IM2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_IM2

FUNCTION FM_INTERVAL_ZM2(MA)
   USE FMVALS
   IMPLICIT NONE
   TYPE (ZM), DIMENSION(:,:) :: MA
   TYPE (FM_INTERVAL), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_ZM2
   INTEGER :: J,K
   INTENT (IN) :: MA
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   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,FM_INTERVAL_ZM2(J,K))
      ENDDO
   ENDDO
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION FM_INTERVAL_ZM2

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

FUNCTION INTERVAL_FM_I(IVAL1,IVAL2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_I
   INTEGER :: IVAL1,IVAL2,IV1,IV2
   INTENT (IN) :: IVAL1,IVAL2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   IV1 = MIN(IVAL1,IVAL2)
   IV2 = MAX(IVAL1,IVAL2)
   CALL FMI2M(IV1,INTERVAL_FM_I%LEFT)
   CALL FMI2M(IV2,INTERVAL_FM_I%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION INTERVAL_FM_I

FUNCTION INTERVAL_FM_R(R1,R2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_R
   REAL :: R1,R2,RV1,RV2
   INTENT (IN) :: R1,R2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   RV1 = MIN(R1,R2)
   RV2 = MAX(R1,R2)
   CALL FMSP2M(RV1,INTERVAL_FM_R%LEFT)
   CALL FMSP2M(RV2,INTERVAL_FM_R%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION INTERVAL_FM_R

FUNCTION INTERVAL_FM_D(D1,D2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_D
   DOUBLE PRECISION :: D1,D2,DV1,DV2
   INTENT (IN) :: D1,D2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   DV1 = MIN(D1,D2)
   DV2 = MAX(D1,D2)
   CALL FMDP2M(DV1,INTERVAL_FM_D%LEFT)
   CALL FMDP2M(DV2,INTERVAL_FM_D%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION INTERVAL_FM_D

FUNCTION INTERVAL_FM_Z(Z1,Z2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_Z
   COMPLEX :: Z1,Z2
   REAL :: RV1,RV2
   INTENT (IN) :: Z1,Z2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   RV1 = MIN(REAL(Z1),REAL(Z2))
   RV2 = MAX(REAL(Z1),REAL(Z2))
   CALL FMSP2M(RV1,INTERVAL_FM_Z%LEFT)
   CALL FMSP2M(RV2,INTERVAL_FM_Z%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION INTERVAL_FM_Z

FUNCTION INTERVAL_FM_ZD(C1,C2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_ZD
   COMPLEX (KIND(0.0D0)) :: C1,C2
   DOUBLE PRECISION :: DV1,DV2
   INTENT (IN) :: C1,C2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   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,INTERVAL_FM_ZD%LEFT)
   CALL FMDP2M(DV2,INTERVAL_FM_ZD%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
END FUNCTION INTERVAL_FM_ZD

FUNCTION INTERVAL_FM_FM(M1,M2)
   USE FMVALS
   IMPLICIT NONE
   TYPE (FM_INTERVAL) :: INTERVAL_FM_FM
   TYPE (FM) :: M1,M2
   INTENT (IN) :: M1,M2
   TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
   CALL FM_UNDEF_INP(M1)
   CALL FM_UNDEF_INP(M2)
   CALL FMMIN(M1%MFM,M2%MFM,MTFM)
   CALL FMMAX(M1%MFM,M2%MFM,MUFM)
   CALL FMEQ(MTFM,INTERVAL_FM_FM%LEFT)
   CALL FMEQ(MUFM,INTERVAL_FM_FM%RIGHT)
   TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION INTERVAL_FM_FM

   FUNCTION INTERVAL_FM_IM(M1,M2)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: INTERVAL_FM_IM
      TYPE (IM) :: M1,M2
      INTENT (IN) :: M1,M2
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_UNDEF_INP(M1)
      CALL FM_UNDEF_INP(M2)
      CALL IMMIN(M1%MIM,M2%MIM,MTIM)
      CALL IMMAX(M1%MIM,M2%MIM,MUIM)
      CALL IMI2FM(MTIM,INTERVAL_FM_IM%LEFT)
      CALL IMI2FM(MUIM,INTERVAL_FM_IM%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION INTERVAL_FM_IM

   FUNCTION INTERVAL_FM_ZM(M1,M2)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: INTERVAL_FM_ZM
      TYPE (ZM) :: M1,M2
      INTENT (IN) :: M1,M2
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_UNDEF_INP(M1)
      CALL FM_UNDEF_INP(M2)
      CALL ZMREAL(M1%MZM,M1FM)
      CALL ZMREAL(M2%MZM,M2FM)
      CALL FMMIN(M1FM,M2FM,INTERVAL_FM_ZM%LEFT)
      CALL FMMAX(M1FM,M2FM,INTERVAL_FM_ZM%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION INTERVAL_FM_ZM

   FUNCTION INTERVAL_FM_ST(S1,S2)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: INTERVAL_FM_ST
      CHARACTER(*) :: S1,S2
      INTENT (IN) :: S1,S2
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMST2M(S1,M1FM)
      CALL FMST2M(S2,M2FM)
      CALL FMMIN(M1FM,M2FM,INTERVAL_FM_ST%LEFT)
      CALL FMMAX(M1FM,M2FM,INTERVAL_FM_ST%RIGHT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION INTERVAL_FM_ST

!                                                               LEFT_ENDPOINT

   FUNCTION LEFT_ENDPOINT_INTERVAL_FM(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: LEFT_ENDPOINT_INTERVAL_FM
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMEQ(MA%LEFT,LEFT_ENDPOINT_INTERVAL_FM%MFM)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION LEFT_ENDPOINT_INTERVAL_FM

!                                                               RIGHT_ENDPOINT

   FUNCTION RIGHT_ENDPOINT_INTERVAL_FM(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: RIGHT_ENDPOINT_INTERVAL_FM
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FMEQ(MA%RIGHT,RIGHT_ENDPOINT_INTERVAL_FM%MFM)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION RIGHT_ENDPOINT_INTERVAL_FM

!                                                               TO_FM

   FUNCTION FM_FM_INTERVAL(MA)

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

      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      TYPE (FM) :: FM_FM_INTERVAL
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMSUB(MA%RIGHT,MA%LEFT,MTFM)
      CALL FMDIVI_R1(MTFM,2)
      CALL FMADD(MA%LEFT,MTFM,FM_FM_INTERVAL%MFM)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_FM_INTERVAL

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

   FUNCTION FM_FM_INTERVAL2(MA)
```

```
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (FM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_FM_INTERVAL2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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,MTFM)
            CALL FMDIVI_R1(MTFM,2)
            CALL FMADD(MA(J,K)%LEFT,MTFM,FM_FM_INTERVAL2(J,K)%MFM)
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_FM_INTERVAL2

!                                                               TO_IM

   FUNCTION IM_FM_INTERVAL(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (IM) :: IM_FM_INTERVAL
      TYPE (FM_INTERVAL) :: MA
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL IMFM2I_INTERVAL(MA,IM_FM_INTERVAL%MIM)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION IM_FM_INTERVAL

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

   FUNCTION IM_FM_INTERVAL2(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (IM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: IM_FM_INTERVAL2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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),IM_FM_INTERVAL2(J,K)%MIM)
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION IM_FM_INTERVAL2

!                                                              TO_ZM

   FUNCTION ZM_FM_INTERVAL(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (ZM) :: ZM_FM_INTERVAL
      TYPE (FM_INTERVAL) :: MA
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMI2M_INTERVAL(0,MUFM_I)
      CALL ZMCMPX_INTERVAL(MA,MUFM_I,ZM_FM_INTERVAL)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION ZM_FM_INTERVAL

   FUNCTION ZM_FM_INTERVAL1(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:) :: MA
      TYPE (ZM), DIMENSION(SIZE(MA)) :: ZM_FM_INTERVAL1
      INTEGER :: J,N
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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,ZM_FM_INTERVAL1(J))
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION ZM_FM_INTERVAL1

   FUNCTION ZM_FM_INTERVAL2(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      TYPE (ZM), DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: ZM_FM_INTERVAL2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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,ZM_FM_INTERVAL2(J,K))
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION ZM_FM_INTERVAL2

!                                                                 TO_INT

   FUNCTION FM_INTERVAL_2INT(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      INTEGER :: FM_INTERVAL_2INT
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2I_INTERVAL(MA,FM_INTERVAL_2INT)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2INT

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

   FUNCTION FM_INTERVAL_2INT2(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      INTEGER, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_2INT2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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),FM_INTERVAL_2INT2(J,K))
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2INT2

!                                                                  TO_SP

   FUNCTION FM_INTERVAL_2SP(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      REAL :: FM_INTERVAL_2SP
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2SP_INTERVAL(MA,FM_INTERVAL_2SP)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2SP

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

   FUNCTION FM_INTERVAL_2SP2(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      REAL, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_2SP2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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),FM_INTERVAL_2SP2(J,K))
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2SP2

!                                                               TO_DP

   FUNCTION FM_INTERVAL_2DP(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      DOUBLE PRECISION :: FM_INTERVAL_2DP
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2DP_INTERVAL(MA,FM_INTERVAL_2DP)
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2DP

   FUNCTION FM_INTERVAL_2DP1(MA)
```

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

   FUNCTION FM_INTERVAL_2DP2(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL), DIMENSION(:,:) :: MA
      DOUBLE PRECISION, DIMENSION(SIZE(MA,DIM=1),SIZE(MA,DIM=2)) :: FM_INTERVAL_2DP2
      INTEGER :: J,K
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      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),FM_INTERVAL_2DP2(J,K))
         ENDDO
      ENDDO
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2DP2

!                                                             TO_SPZ

   FUNCTION FM_INTERVAL_2SPZ(MA)
      USE FMVALS
      IMPLICIT NONE
      TYPE (FM_INTERVAL) :: MA
      COMPLEX FM_INTERVAL_2SPZ
      REAL :: R
      INTENT (IN) :: MA
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      CALL FM_INTERVAL_UNDEF_INP(MA)
      CALL FMM2SP_INTERVAL(MA,R)
      FM_INTERVAL_2SPZ = CMPLX( R , 0.0 )
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
   END FUNCTION FM_INTERVAL_2SPZ

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