```fortran
   MODULE FMVALS

!  These are the global and saved variables used by the FM package.
!  See the FM_User_Manual.txt file for further description of some of these variables.

!  They are initialized assuming the program will run on a 32-bit computer with variables in
!  FM.f95 having names beginning with 'M' being declared as having 64-bit representations
!  (DOUBLE PRECISION).

!  For a machine with a different architecture, or for setting the precision level to a different
!  value, CALL FMSET(NPREC) before doing any multiple precision operations.  FMSET tries to
!  initialize the variables to the best values for the given machine.  To have the values chosen
!  by FMSET written on unit KW, CALL FMVARS.

!  Base and precision will be set to give slightly more than 50 decimal digits of precision, giving
!  the user 50 significant digits of precision along with several base ten guard digits.

!  MBASE is set to 10**7.
!  JFORM1 and JFORM2 are set to ES format displaying 50 significant digits.

!  The trace option is set off.
!  The mode for angles in trig functions is set to radians.
!  The rounding mode is set to symmetric rounding (to nearest).
!  Warning error message level is set to 1.
!  Cancellation error monitor is set off.
!  Screen width for output is set to 80 columns.
!  The exponent character for FM output is set to 'M'.
!  Debug error checking is set off.

!  KW, the unit number for all FM output, is set to 6.

      PRIVATE AINT, CEILING, DIGITS, EPSILON, HUGE, INT, LOG, MAX, MIN, SQRT

      REAL (KIND(1.0D0)), PARAMETER :: M_TWO = 2
      DOUBLE PRECISION, PARAMETER :: DP_TWO = 2
      INTEGER, PARAMETER :: I_TWO = 2
      REAL, PARAMETER :: R_TWO = 2

!             KW is the unit number for standard output from the FM package.
!                This includes trace output and error messages.

      INTEGER, SAVE :: KW = 6

!             The min below is needed when m-variables have more precision than double,
!             as with 64-bit integer m-variables and 64-bit doubles (53-bit precision).

      REAL (KIND(1.0D0)), PARAMETER :: MAX_REPRESENTABLE_M_VAR =  &
                          ( (M_TWO ** (MIN(DIGITS(M_TWO),DIGITS(DP_TWO))-1)) - 1 ) * 2 + 1

!             MAXINT should be set to a very large integer, possibly the largest representable
!                    integer for the current machine.  For most 32-bit machines, MAXINT is set
!                    to  2**53 - 1 = 9.007D+15  when double precision arithmetic is used for
!                    M-variables.  Using integer M-variables usually gives
!                    MAXINT = 2**31 - 1 = 2147483647.
!                    Setting MAXINT to a smaller number is ok, but this unnecessarily restricts
!                    the permissible range of MBASE and MXEXP.

      REAL (KIND(1.0D0)), SAVE :: MAXINT = MAX_REPRESENTABLE_M_VAR

!             INTMAX is a large value close to the overflow threshold for integer variables.
!                    It is usually 2**31 - 1 for machines with 32-bit integer arithmetic.

      INTEGER, SAVE :: INTMAX = HUGE(I_TWO)

!             DPMAX should be set to a value near the machine's double precision overflow threshold,
!                   so that DPMAX and 1.0D0/DPMAX are both representable in double precision.

      DOUBLE PRECISION, SAVE :: DPMAX = HUGE(DP_TWO)/5

!             SPMAX should be set to a value near the machine's single precision overflow threshold,
!                   so that 1.01*SPMAX and 1.0/SPMAX are both representable in single precision.

      REAL, SAVE :: SPMAX = HUGE(R_TWO)/5

!             MXBASE is the maximum value for MBASE.

      REAL (KIND(1.0D0)), PARAMETER :: MAX_BASE = AINT(SQRT(MAX_REPRESENTABLE_M_VAR + 1.0D-9))

      REAL (KIND(1.0D0)), SAVE :: MXBASE = MAX_BASE

!             MBASE is the currently used base for arithmetic.

      REAL (KIND(1.0D0)), PARAMETER :: M_TEN = 10

      REAL (KIND(1.0D0)), SAVE :: MBASE = M_TEN ** AINT(LOG(MAX_BASE/4.0D0) / LOG(10.0D0))

!             NDIG is the number of digits currently being carried.

      INTEGER, SAVE :: NDIG = CEILING( 52.0D0 / AINT(LOG(MAX_BASE/4.0D0)/LOG(10.0D0)) ) + 1

!             KFLAG is the flag for error conditions.

      INTEGER, SAVE :: KFLAG = 0

!             NTRACE is the trace switch.  Default is no printing.

      INTEGER, SAVE :: NTRACE = 0

!             LVLTRC is the trace level.  Default is to trace only routines called directly
!                    by the user.

      INTEGER, SAVE :: LVLTRC = 1

!             NCALL is the call stack pointer.

      INTEGER, SAVE :: NCALL = 0

!             RAISE_NDIG is set to 1 when one FM routine calls another and the second one needs
!                        to use more guard digits.

      INTEGER, SAVE :: RAISE_NDIG = 0
```

```
!             NAMEST is the call stack.

      INTEGER, PRIVATE :: I
      CHARACTER(9), SAVE :: NAMEST(0:50) = (/ ('MAIN     ' , I = 0, 50) /)

!             Some constants that are often needed are stored with the maximum precision to which
!             they have been computed in the currently used base.  This speeds up the trig, log,
!             power, and exponential functions.

!             NDIGPI is the number of digits available in the currently stored value of pi (MPISAV).

      INTEGER, SAVE :: NDIGPI = 0

!             MBSPI is the value of MBASE for the currently stored value of pi.

      REAL (KIND(1.0D0)), SAVE :: MBSPI = 0

!             NDIGE is the number of digits available in the currently stored value of e (MESAV).

      INTEGER, SAVE :: NDIGE = 0

!             MBSE is the value of MBASE for the currently stored value of e.

      REAL (KIND(1.0D0)), SAVE :: MBSE = 0

!             NDIGLB is the number of digits available in the currently stored value of LN(MBASE)
!                    (MLBSAV).

      INTEGER, SAVE :: NDIGLB = 0

!             MBSLB is the value of MBASE for the currently stored value of LN(MBASE).

      REAL (KIND(1.0D0)), SAVE :: MBSLB = 0

!             NDIGLI is the number of digits available in the currently stored values of the four
!                    logarithms used by FMLNI:  MLN2, MLN3, MLN5, MLN7.

      INTEGER, SAVE :: NDIGLI = 0

!             MBSLI is the value of MBASE for the currently stored values of MLN2, MLN3, MLN5, MLN7.

      REAL (KIND(1.0D0)), SAVE :: MBSLI = 0

!             MXEXP  is the current maximum exponent.
!             MXEXP2 is the internal maximum exponent.  This is used to define the overflow and
!                    underflow thresholds.
!
!             These values are chosen so that FM routines can raise the overflow/underflow limit
!             temporarily while computing intermediate results.  MXEXP2 satisfies these conditions:
!             1. EXP(INTMAX) > MXBASE**(MXEXP2+1)
!             2. MXEXP2 < MAXINT/20
!
!             The overflow threshold is MBASE**(MXEXP+1), and the underflow threshold is
!             MBASE**(-MXEXP-1).  This means the valid exponents in the first word of an FM
!             number can range from -MXEXP to MXEXP+1 (inclusive).
```

```
      REAL (KIND(1.0D0)), PARAMETER :: MAX_EXPONENT = AINT( MIN(                                  &
                                       MAX(HUGE(INTMAX) / LOG(MAX_BASE+1.0D-9) , 117496405.0D0),   &
                                       MAX_REPRESENTABLE_M_VAR / 20.0D0) )

      REAL (KIND(1.0D0)), SAVE :: MXEXP = AINT( MAX_EXPONENT / 2.01D0 + 0.5D0 )

      REAL (KIND(1.0D0)), SAVE :: MXEXP2 = MAX_EXPONENT

!             KACCSW is a switch used to enable cancellation error monitoring.  Routines where
!                    cancellation is not a problem run faster by skipping the cancellation monitor
!                    calculations.
!                    KACCSW = 0 means no error monitoring,
!                           = 1 means error monitoring is done.

      INTEGER, SAVE :: KACCSW = 0

!             MEXPUN is the exponent used as a special symbol for underflowed results.

      REAL (KIND(1.0D0)), SAVE :: MEXPUN = AINT( -MAX_EXPONENT * 1.01D0 )

!             MEXPOV is the exponent used as a special symbol for overflowed results.

      REAL (KIND(1.0D0)), SAVE :: MEXPOV = AINT( MAX_EXPONENT * 1.01D0 )

!             MUNKNO is the exponent used as a special symbol for unknown FM results
!                    (1/0, SQRT(-3.0), ...).  When changing this value, also change the three
!                    TYPE FM, IM, ZM initializations in FMZM90.f95.

      REAL (KIND(1.0D0)), SAVE :: MUNKNO = AINT( MAX_EXPONENT * 1.0201D0 )

!             RUNKNO is returned from FM to real or double conversion routines when no valid result
!                    can be expressed in real or double precision.  On systems that provide a value
!                    for undefined results (e.g., Not A Number) setting RUNKNO to that value is
!                    reasonable.  On other systems set it to a value that is likely to make any
!                    subsequent results obviously wrong that use it.  In either case a KFLAG = -4
!                    condition is also returned.

      REAL, SAVE :: RUNKNO = -1.01*(HUGE(R_TWO)/3.0)

!             IUNKNO is returned from FM to integer conversion routines when no valid result can be
!                    expressed as a one word integer.  KFLAG = -4 is also set.

      INTEGER, SAVE :: IUNKNO = -HUGE(I_TWO)/18

!             JFORM1 indicates the format used by FMOUT.

      INTEGER, SAVE :: JFORM1 = 1

!             JFORM2 indicates the number of digits used in FMOUT.

      INTEGER, SAVE :: JFORM2 = 50

!             KRAD = 1 indicates that trig functions use radians,
!                  = 0 means use degrees.
```

```fortran
      INTEGER, SAVE :: KRAD = 1

!             KWARN = 0 indicates that no warning message is printed and execution continues when
!                       UNKNOWN or another exception is produced.
!                   = 1 means print a warning message and continue.
!                   = 2 means print a warning message and stop.

      INTEGER, SAVE :: KWARN = 1

!             KROUND =  1  causes all results to be rounded to the nearest FM number, or to the
!                          value with an even last digit if the result is halfway between two
!                          FM numbers.
!                    =  0  causes all results to be rounded toward zero (chopped).
!                    = -1  causes all results to be rounded toward minus infinity.
!                    =  2  causes all results to be rounded toward plus infinity.

      INTEGER, SAVE :: KROUND = 1

!             KRPERF = 1  causes more guard digits to be used, to get perfect rounding in the mode
!                         set by KROUND.
!                    = 0  causes a smaller number of guard digits used, to give nearly perfect
!                         rounding.  This number is chosen so that the last intermediate result
!                         should have error less than 0.001 unit in the last place of the final
!                         rounded result.
!             Beginning with version 1.3 KRPERF is not used, since perfect rounding is always done.
!             The variable has been left in the package for compatibility with earlier versions.

      INTEGER, SAVE :: KRPERF = 0

!             KSWIDE defines the maximum screen width to be used for all unit KW output.

      INTEGER, SAVE :: KSWIDE = 80

!             KESWCH = 1   causes input to FMINP with no digits before the exponent letter to be
!                          treated as if there were a leading '1'.  This is sometimes better for
!                          interactive input:  'E7' converts to 10.0**7.
!                    = 0   causes a leading zero to be assumed.  This gives compatibility with
!                          Fortran:  'E7' converts to 0.0.

      INTEGER, SAVE :: KESWCH = 1

!             CMCHAR defines the exponent letter to be used for FM variable output from FMOUT,
!                    as in 1.2345M+678.
!                    Change it to 'E' for output to be read by a non-FM program.

      CHARACTER, SAVE :: CMCHAR = 'M'

!             KDEBUG = 0   Error checking is not done for valid input arguments and parameters
!                          like NDIG and MBASE upon entry to each routine.
!                    = 1   Error checking is done.

      INTEGER, SAVE :: KDEBUG = 0

!             KROUND_RETRY is an internal flag controlling cases where the result has close to
!                          1/2 ulp of error and the operation should be done again with more
!                          guard digits to insure perfect rounding.
```

```
      INTEGER, SAVE :: KROUND_RETRY = 0

!             KSUB is an internal flag set during subtraction so that the addition routine will
!                  negate its second argument.

      INTEGER, SAVE :: KSUB = 0

!             KSQR is an internal flag set during squaring so that at high precision the
!                  multiplication routine will not need to compute the fft of its second argument.

      INTEGER, SAVE :: KSQR = 0

!             KREM is an internal flag set during high precision integer division operations to
!                  indicate that the remainder in IMDIVR need not be computed.

      INTEGER, SAVE :: KREM = 1

!             JRSIGN is an internal flag set during arithmetic operations so that the rounding
!                    routine will know the sign of the final result.

      INTEGER, SAVE :: JRSIGN = 0

!             LHASH is a flag variable used to indicate when to initialize two hash tables that are
!                   used for character look-up during input conversion.
!                   LHASH = 1 means that the tables have been built.
!             LHASH1 and LHASH2 are the array dimensions of the tables.
!             KHASHT and KHASHV are the two tables.

      INTEGER, SAVE :: LHASH = 0
      INTEGER, PARAMETER :: LHASH1 =   0
      INTEGER, PARAMETER :: LHASH2 = 256
      INTEGER, SAVE :: KHASHT(LHASH1:LHASH2),KHASHV(LHASH1:LHASH2)

!             DPEPS is the approximate machine precision.

      DOUBLE PRECISION, SAVE :: DPEPS = EPSILON(DP_TWO)

!             LJSUMS is the maximum number of concurrent sums to use in function evaluation.

      INTEGER, PARAMETER :: LJSUMS = 1000

!             Saved constants that depend on MBASE.

      REAL (KIND(1.0D0)), SAVE :: MBLOGS = 0
!             (Setting MBLOGS to zero here will cause the other variables that depend on MBASE
!              to automatically be defined when the first FM operation is done.)

      REAL, SAVE :: ALOGMB = 1.611810E+1
      REAL, SAVE :: ALOGM2 = 2.325350E+1
      REAL, SAVE :: ALOGMX = 3.673680E+1
      REAL, SAVE :: ALOGMT = 7.0E0

      INTEGER, SAVE :: NGRD21 = 1
      INTEGER, SAVE :: NGRD52 = 2
      INTEGER, SAVE :: NGRD22 = 2
```

```fortran
      REAL (KIND(1.0D0)), SAVE :: MEXPAB = AINT(MAX_EXPONENT / 5.0D0)

      DOUBLE PRECISION, SAVE :: DLOGMB = 1.611809565095832D+1
      DOUBLE PRECISION, SAVE :: DLOGTN = 2.302585092994046D+0
      DOUBLE PRECISION, SAVE :: DLOGTW = 6.931471805599453D-1
      DOUBLE PRECISION, SAVE :: DPPI   = 3.141592653589793D+0
      DOUBLE PRECISION, SAVE :: DLOGTP = 1.837877066409345D+0
      DOUBLE PRECISION, SAVE :: DLOGPI = 1.144729885849400D+0
      DOUBLE PRECISION, SAVE :: DLOGEB = 2.236222824932432D+0

      REAL (KIND(1.0D0)), SAVE :: MBASEL = 0
      REAL (KIND(1.0D0)), SAVE :: MBASEN = 0
      REAL (KIND(1.0D0)), SAVE :: M_VAL, M_VAL1, M_VAL2, M_VAL3, M_VAL4

      INTEGER, SAVE :: NDIGL = 0
      INTEGER, SAVE :: NDIGN = 0
      INTEGER, SAVE :: NGUARL = 0
      INTEGER, SAVE :: N21
      INTEGER, SAVE :: NGRDN

!             These variables are used by FM_RANDOM_NUMBER.
!             MBRAND is the base used for the random number arithmetic.
!                    It needs to remain the same even if the user changes MBASE.

      REAL (KIND(1.0D0)), SAVE :: MBRAND = M_TEN ** AINT(LOG(MAX_BASE/4.0D0) / LOG(10.0D0))

      INTEGER, SAVE :: MRNX = -3
      INTEGER, SAVE :: MRNA = -3
      INTEGER, SAVE :: MRNM = -3
      INTEGER, SAVE :: MRNC = -3
      INTEGER, SAVE :: START_RANDOM_SEQUENCE = -1
      INTEGER, SAVE :: LAST_DIGIT_OF_M_M1
      DOUBLE PRECISION, SAVE :: DPM

!             Work area for FM numbers, and related variables.

      INTEGER, SAVE :: SIZE_OF_MWK = 0
      REAL (KIND(1.0D0)), SAVE, DIMENSION(:), ALLOCATABLE :: MWK, MOVE_MWK, MOVE_F
      INTEGER, PARAMETER :: START_RESIZE  = 100000
      INTEGER, SAVE :: SIZE_OF_START = 2 * START_RESIZE
      LOGICAL, SAVE :: IN_USER_FUNCTION = .FALSE.
      INTEGER, SAVE :: USER_FUNCTION_LEVEL = 0
      INTEGER, SAVE :: LEVEL_OF_RECURSION = 0
      INTEGER, SAVE :: NUMBER_USED_AT_LEVEL(1000)
      INTEGER, SAVE, DIMENSION(:), ALLOCATABLE :: START, TEMPV, RESIZE, SIZE_OF, TEMP7
      INTEGER, PARAMETER :: SIZE_OF_TEMP6 = 100
      INTEGER, SAVE :: FMTEMP6(SIZE_OF_TEMP6), NMAX_FMTEMP6 = 0, N_FMTEMP6 = 0, TOTAL_FMTEMP6 = 0
      INTEGER, SAVE :: IMTEMP6(SIZE_OF_TEMP6), NMAX_IMTEMP6 = 0, N_IMTEMP6 = 0, TOTAL_IMTEMP6 = 0
      INTEGER, SAVE :: TOTAL_TEMP7 = 0
      INTEGER, SAVE :: LOWEST_SAVED_AREA_INDEX = 2*START_RESIZE + 1
      INTEGER, SAVE :: START_OF_MWK_SAVED_AREA = 0
      INTEGER, SAVE :: MINIMUM_SAVED_CONSTANTS_USED = 10**9
      INTEGER, SAVE :: NUMBER_USED = 0
      INTEGER, SAVE :: MAXIMUM_NUMBER_USED = 0
      INTEGER, SAVE :: MAXIMUM_MWK_USED = 0
      INTEGER, SAVE :: RESULT_SIZE = 0
      INTEGER, SAVE :: TEMPV_CALL_STACK = 0
      INTEGER, SAVE :: MWA = -4
      INTEGER, SAVE :: MWD = -4
      INTEGER, SAVE :: MWE = -4
      INTEGER, SAVE :: MPA = -3
      INTEGER, SAVE :: MPB = -3
      INTEGER, SAVE :: MPC = -3
      INTEGER, SAVE :: MPD = -3
      INTEGER, SAVE :: MWI = -3
      INTEGER, SAVE :: MPMA = -3
      INTEGER, SAVE :: MPMB = -3
      INTEGER, SAVE :: MPX(2) = (/ -3, -3 /)
      INTEGER, SAVE :: MPY(2) = (/ -3, -3 /)
      INTEGER, SAVE :: MPZ(2) = (/ -3, -3 /)

!             Variables related to input/output and formatting.

      INTEGER, SAVE :: LMBUFF = 0
      INTEGER, SAVE :: LMBUFZ = 0
      CHARACTER, SAVE, DIMENSION(:), ALLOCATABLE :: CMBUFF,CMBUFZ,MOVE_CMBUFF

!             Saved FM constants.

      INTEGER, SAVE :: MPISAV = -3
      INTEGER, SAVE :: MESAV = -3
      INTEGER, SAVE :: MLBSAV = -3
      INTEGER, SAVE :: MLN2 = -3
      INTEGER, SAVE :: MLN3 = -3
      INTEGER, SAVE :: MLN5 = -3
      INTEGER, SAVE :: MLN7 = -3

!             Set the default value of JFORMZ to give ' 1.23 + 4.56 i ' style format for output
!             of complex variables.

      INTEGER, SAVE :: JFORMZ = 1

!             Set the default value of JPRNTZ to print real and imaginary parts on one line
!             whenever possible.

      INTEGER, SAVE :: JPRNTZ = 1

!             MBERN  is the array used to save Bernoulli numbers so they do not have to be
!                    re-computed on subsequent calls.
!             NDBERN is the array used to save the number of digits in the current base for
!                    each of the saved Bernoulli numbers.

!             MBSBRN is the value of MBASE for the currently saved Bernoulli numbers.

      REAL (KIND(1.0D0)), SAVE :: MBSBRN = 0

!             NUMBRN is the number of the largest Bernoulli number saved using base MBSBRN.

      INTEGER, SAVE :: NUMBRN = 0

!             LMBERN is the size of the arrays MBERN and NDBERN.
```

```
      INTEGER, PARAMETER :: LMBERN = 60000
      INTEGER, SAVE, DIMENSION(LMBERN) :: MBERN = (/ (-3 , I = 1, LMBERN) /)
      INTEGER, SAVE, DIMENSION(LMBERN) :: NDBERN = 0

!             B(2N) is stored in MBERN(N) for 2N >= 28.

!             M_EULER     is the saved value of Euler's constant.
!             M_GAMMA_MA  is the last input value to FMGAM, and
!             M_GAMMA_MB  is the corresponding output value.
!             M_LN_2PI    holds the saved value of LN(2*pi).

    INTEGER, SAVE :: M_EULER = -3
    INTEGER, SAVE :: M_GAMMA_MA = -3
    INTEGER, SAVE :: M_GAMMA_MB = -3
    INTEGER, SAVE :: M_LN_2PI = -3

!             MBSGAM is the value of MBASE used in the currently stored value of
!                    M_GAMMA_MA and M_GAMMA_MB.
!             NDGGAM is the maximum NDIG used in the currently stored value of
!                    M_GAMMA_MA and M_GAMMA_MB.

      REAL (KIND(1.0D0)), SAVE :: MBSGAM = 0

      INTEGER, SAVE :: NDGGAM = 0

!             MBS2PI is the value of MBASE used in the currently stored value of LN(2*pi).
!             NDG2PI is the maximum NDIG used in the currently stored value of LN(2*pi).

      REAL (KIND(1.0D0)), SAVE :: MBS2PI = 0

      INTEGER, SAVE :: NDG2PI = 0

!             MBSEUL is the value of MBASE used in the currently stored value of M_EULER.
!             NDGEUL is the maximum NDIG used in the currently stored value of M_EULER.

      REAL (KIND(1.0D0)), SAVE :: MBSEUL = 0

      INTEGER, SAVE :: NDGEUL = 0

   END MODULE FMVALS
```