This file extends the FM_User_Manual.txt file for the files that support the FM_parallel package. The parallel versions of 4 files from the standard version of FM are included in one file for the parallel version: FMSAVE.f95, FM.f95, FMZM90.f95, and FM_Sample_Routines.f95.

This is the thread-safe version of FM. It can be used with a program that uses coarrays, which is the Fortran-standard way of parallel programming. It should also be ok to use with other, non-standard, ways of parallel programming (openMP, MPI, etc.).

Because of restrictions imposed by the need to be thread-safe, this version of FM has several limitations compared to the regular version of FM.

1. Global variables defined in modules are not thread-safe if they can be changed while the program runs. This means the user sets the FM precision level by defining variable FM_SIGNIFICANT_DIGITS here in module FMVALS_PARALLEL, then compiles this file and links it to their program. Calling FM_SET to set the precision is not available.

2. TYPE(IM) integer multi-precision numbers have varying numbers of digits that are indirectly limited by FM_SIGNIFICANT_DIGITS. Make sure that all TYPE(IM) values, intermediate as well as final results, have fewer than 20*FM_SIGNIFICANT_DIGITS decimal digits.

3. It is less common, but if a user program wants different values for other FM options like rounding mode, screen width for FM output, etc., they must be initialized here in this module and not changed while the program runs.

4. Similarly, FM precision level cannot be changed by the user's program while it runs.

5. The FM_RANDOM_NUMBER random number generator cannot be used, since it depends on a global saved state to get the next number.

6. Because multi-precision variables are no longer stored in a global module database, the FM_DEALLOCATE function is not needed and has been removed. Similarly, the FM_(ENTER or EXIT)_USER_(FUNCTION or ROUTINE) calls are not needed and those routines have been removed.

7. Saved values like pi, e, euler gamma, etc., are global variables in the standard version of FM. They have been removed from this thread-safe version, so they are re-computed each time they are needed. That makes some functions like trig and log/exponential functions slightly slower.

8. The global allocatable database from version 1.3 has been replaced with local fixed-size arrays for the multiple precision numbers. A few routines like fm_fprime and zm_fprime that relied on raising precision a lot to overcome unstable formulas may now return unknown. Up to 4th or 5th derivatives should usually be ok, but higher derivatives may fail.

These are the files included for the parallel FM package.

1. FM_parallel.f95   The routines and interfaces for parallel operations

2. SampleFM_parallel_openmp.f95   Sample program using openmp.

3. SampleFM_parallel_coarray.f95   Sample program using coarray.

4. TestFM_parallel_openmp.f95   Test program using openmp.

5. TestFM_parallel_coarray.f95   Test program using coarray.

There are several different ways to get multi-threaded parallel execution, and not all compilers support all forms. The basic FM_parallel.f95 file is the same in all cases, since the commands to control parallel computations are contained in the programs that use it.

I have tested FM_parallel using openmp or coarrays on three compilers:

gfortran with openmp
ifort with openmp
nagfor with coarrays

Where a program using the standard FM has USE FMZM to access the FM arithmetic, the parallel version has USE FMZM_PARALLEL.

In the parallel version the variables themselves in the user's program are declared the same way as before, TYPE (FM), etc.

Here are sample commands to run the program in file 2. These are for the gfortran compiler on a Mac. PCs are very similar, as are other compilers. See FM_User_Manual.txt.

Multi-threaded applications seem to be more fragile than single-thread versions. Some compilers may need extra options. On my machine I need to increase the stack size for each thread to get openmp to work properly.

```
export OMP_STACKSIZE=500m

gfortran  FM_parallel.f95  -O3  -fopenmp  -c

gfortran  SampleFM_parallel_openmp.f95  -O3  -fopenmp  -c

gfortran  FM_parallel.o  SampleFM_parallel_openmp.o  -fopenmp  -o SampleFM_parallel_openmp

./SampleFM_parallel_openmp
```

Here are sample commands to run the program in file 5. These are for the nagfor compiler on a Mac. Using optimization level -O3 is ok with nagfor for SampleFM_parallel_coarray, but -O2 -Wc,-O1 is needed for TestFM_parallel_coarray. As of 2021 the nagfor developers said the problem with -O3 seemed to be a bug in the clang (back-end C compiler) optimizer, and only showed up on Intel-based machines.

```
nagfor FM_parallel.f95  -O2 -Wc,-O1  -coarray  -c

nagfor TestFM_parallel_coarray.f95  -O2 -Wc,-O1  -coarray  -c

nagfor FM_parallel.o TestFM_parallel_coarray.o  -coarray  -o TestFM_parallel_coarray

./TestFM_parallel_coarray
```