```
      SUBROUTINE FM_LIN_SOLVE(A,X,B,N,DET)
      USE FMVALS
      USE FMZM
      IMPLICIT NONE

!  Gauss elimination to solve the linear system  A X = B, where:

!  A   is the matrix of the system, containing the  N x N coefficient matrix.

!  B   is the  N x 1  right-hand-side vector.

!  X   is the returned  N x 1  solution vector.

!  DET is returned as the determinant of A.
!      Nonzero DET means a solution was found.
!      DET = 0 is returned if the system is singular.

!  A,X,B,DET are all type (fm) multiprecision variables.

      INTEGER :: N
      TYPE (FM) :: A(N,N), B(N), X(N), DET
      TYPE (FM), SAVE :: TOL
      TYPE (FM), ALLOCATABLE :: A1(:,:), A2(:,:), B1(:), R1(:), X1(:)
      INTEGER, ALLOCATABLE :: KSWAP(:)
      INTEGER :: I, J, NDSAVE

      CALL FM_ENTER_USER_ROUTINE
      ALLOCATE(A1(N,N),A2(N,N),B1(N),R1(N),X1(N),KSWAP(N),STAT=J)
      IF (J /= 0) THEN
          WRITE (*,"(/' Error in FM_LIN_SOLVE.  Unable to allocate arrays with N = ',I8/)") N
          STOP
      ENDIF

      TOL = EPSILON(B(1))/MBASE/TO_FM(10)**10

      NDSAVE = NDIG
      NDIG = 2*NDIG

!             Copy A and B to A1 and B1 with higher precision.

  110 CALL FM_EQU_R1(TOL,NDSAVE,NDIG)
      DO I = 1, N
         DO J = 1, N
            CALL FM_EQU(A(I,J),A1(I,J),NDSAVE,NDIG)
            CALL FM_EQ(A1(I,J),A2(I,J))
         ENDDO
         CALL FM_EQU(B(I),B1(I),NDSAVE,NDIG)
      ENDDO

!             Solve the system.

      CALL FM_FACTOR_LU(A1,N,DET,KSWAP)
      IF (DET == 0 .OR. IS_UNKNOWN(DET)) THEN
          IF (KWARN > 0) THEN
              WRITE (KW,"(/' Error in FM_LIN_SOLVE.  The matrix is singular.'/)")
```

```
          ENDIF
          IF (KWARN >= 2) STOP
          X1 = TO_FM(' UNKNOWN ')
          GO TO 120
       ENDIF
       CALL FM_SOLVE_LU(A1,N,B1,X1,KSWAP)

!             Do an iterative refinement.

       R1 = MATMUL(A2,X1) - B1

       CALL FM_SOLVE_LU(A1,N,R1,B1,KSWAP)
       X1 = X1 - B1

!             Check for accuracy at the user's precision.

       IF (SQRT( DOT_PRODUCT( B1 , B1 ) ) > TOL) THEN
          NDIG = 2*NDIG
          GO TO 110
       ENDIF

!             Round and return X and DET.

  120  DO I = 1, N
          CALL FM_EQU(X1(I),X(I),NDIG,NDSAVE)
       ENDDO
       CALL FM_EQU_R1(DET,NDIG,NDSAVE)

       NDIG = NDSAVE

       CALL FM_DEALLOCATE(A1)
       CALL FM_DEALLOCATE(A2)
       CALL FM_DEALLOCATE(B1)
       CALL FM_DEALLOCATE(R1)
       CALL FM_DEALLOCATE(X1)
       DEALLOCATE(A1,A2,B1,R1,X1,KSWAP)

       CALL FM_EXIT_USER_ROUTINE
       END SUBROUTINE FM_LIN_SOLVE

       SUBROUTINE FM_FACTOR_LU(A,N,DET,KSWAP)
       USE FMZM
       IMPLICIT NONE

!  Gauss elimination to factor the NxN matrix A (LU decomposition).

!  The time is proportional to  N**3.

!  Once this factorization has been done, a linear system  A x = b
!  with the same coefficient matrix A and Nx1 vector b can be solved
!  for x using routine FM_SOLVE_LU in time proportional to  N**2.

!  DET is returned as the determinant of A.
!      Nonzero DET means there is a unique solution.
!      DET = 0 is returned if the system is singular.
```

```
!  KSWAP is a list of row interchanges made by the partial pivoting strategy during the
!        elimination phase.

!  After returning, the values in matrix A have been replaced by the multipliers
!  used during elimination.  This is equivalent to factoring the A matrix into
!  a lower triangular matrix L times an upper triangular matrix U.

      INTEGER :: N
      INTEGER :: JCOL, JDIAG, JMAX, JROW, KSWAP(N)
      TYPE (FM) :: A(N,N), DET
      TYPE (FM), SAVE :: AMAX, AMULT, TEMP

      CALL FM_ENTER_USER_ROUTINE
      DET = 1
      KSWAP(1:N) = 1
      IF (N <= 0) THEN
          DET = 0
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      IF (N == 1) THEN
          KSWAP(1) = 1
          DET = A(1,1)
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF

!             Do the elimination phase.
!             JDIAG is the current diagonal element below which the elimination proceeds.

      DO JDIAG = 1, N-1

!             Pivot to put the element with the largest absolute value on the diagonal.

         AMAX = ABS(A(JDIAG,JDIAG))
         JMAX = JDIAG
         DO JROW = JDIAG+1, N
            IF (ABS(A(JROW,JDIAG)) > AMAX) THEN
                AMAX = ABS(A(JROW,JDIAG))
                JMAX = JROW
            ENDIF
         ENDDO

!             If AMAX is zero here then the system is singular.

         IF (AMAX == 0.0) THEN
             DET = 0
             CALL FM_EXIT_USER_ROUTINE
             RETURN
         ENDIF

!             Swap rows JDIAG and JMAX unless they are the same row.

         KSWAP(JDIAG) = JMAX
         IF (JMAX /= JDIAG) THEN
             DET = -DET
```

```
            DO JCOL = JDIAG, N
               TEMP = A(JDIAG,JCOL)
               A(JDIAG,JCOL) = A(JMAX,JCOL)
               A(JMAX,JCOL) = TEMP
            ENDDO
         ENDIF
         DET = DET * A(JDIAG,JDIAG)

!             For JROW = JDIAG+1, ..., N, eliminate A(JROW,JDIAG) by replacing row JROW by
!                 row JROW - A(JROW,JDIAG) * row JDIAG / A(JDIAG,JDIAG)

         DO JROW = JDIAG+1, N
            IF (A(JROW,JDIAG) == 0) CYCLE
            AMULT = A(JROW,JDIAG)/A(JDIAG,JDIAG)

!             Save the multiplier for use later by FM_SOLVE_LU.

            A(JROW,JDIAG) = AMULT
            DO JCOL = JDIAG+1, N
               CALL FMMPY_SUB(A(JROW,JCOL)%MFM,AMULT%MFM,A(JDIAG,JCOL)%MFM)
            ENDDO
         ENDDO
      ENDDO
      DET = DET * A(N,N)

      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE FM_FACTOR_LU

      SUBROUTINE FM_SOLVE_LU(A,N,B,X,KSWAP)
      USE FMZM
      IMPLICIT NONE

!  Solve a linear system  A x = b.
!  A is the NxN coefficient matrix, after having been factored by FM_FACTOR_LU.
!  B is the Nx1 right-hand-side vector.
!  X is returned with the solution of the linear system.
!  KSWAP is a list of row interchanges made by the partial pivoting strategy during the
!        elimination phase in FM_FACTOR_LU.
!  Time for this call is proportional to  N**2.

      INTEGER :: N, KSWAP(N)
      TYPE (FM) :: A(N,N), B(N), X(N)
      INTEGER :: J, JDIAG, JMAX
      TYPE (FM), SAVE :: TEMP

      CALL FM_ENTER_USER_ROUTINE
      IF (N <= 0) THEN
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      IF (N == 1) THEN
          X(1) = B(1) / A(1,1)
          CALL FM_EXIT_USER_ROUTINE
          RETURN
      ENDIF
      DO J = 1, N
```

```
         X(J) = B(J)
      ENDDO

!             Do the elimination phase operations only on X.
!             JDIAG is the current diagonal element below which the elimination proceeds.

      DO JDIAG = 1, N-1

!             Pivot to put the element with the largest absolute value on the diagonal.

         JMAX = KSWAP(JDIAG)

!             Swap rows JDIAG and JMAX unless they are the same row.

         IF (JMAX /= JDIAG) THEN
             TEMP = X(JDIAG)
             X(JDIAG) = X(JMAX)
             X(JMAX) = TEMP
         ENDIF

!             For JROW = JDIAG+1, ..., N, eliminate A(JROW,JDIAG) by replacing row JROW by
!                 row JROW - A(JROW,JDIAG) * row JDIAG / A(JDIAG,JDIAG)
!             After factoring, A(JROW,JDIAG) is the original A(JROW,JDIAG) / A(JDIAG,JDIAG).

         DO J = JDIAG+1, N
            X(J) = X(J) - A(J,JDIAG) * X(JDIAG)
         ENDDO
      ENDDO

!             Do the back substitution.

      DO JDIAG = N, 1, -1

!             Divide row JDIAG by the diagonal element.

         X(JDIAG) = X(JDIAG) / A(JDIAG,JDIAG)

!             Zero above the diagonal in column JDIAG by replacing row JROW by
!                 row JROW - A(JROW,JDIAG) * row JDIAG
!             For JROW = 1, ..., JDIAG-1.

         IF (JDIAG == 1) EXIT
         DO J = 1, JDIAG-1
            X(J) = X(J) - A(J,JDIAG) * X(JDIAG)
         ENDDO
      ENDDO

      CALL FM_EXIT_USER_ROUTINE
      END SUBROUTINE FM_SOLVE_LU

      SUBROUTINE FMMPY_SUB(MA,MB,MC)
      USE FMVALS
      IMPLICIT NONE

!  Fused multiply-subtract operation.  Return  MA = MA - MB*MC
!  This is an internal FM routine used by FM_FACTOR_LU.  It doesn't always return correctly
```

```
!  rounded results, since precision will have already been raised by FM_LIN_SOLVE before
!  calling FM_FACTOR_LU.

      INTEGER :: MA,MB,MC
      INTEGER :: J,K,K1,KPTA,KPTC,MXY,NUMBER_USED_SAVE,MA_VS_MBMC
      DOUBLE PRECISION :: A1,A2,B,B1,B2,C1,C2,DPA,DPBC
      REAL (KIND(1.0D0)) :: MBJ, MGD

!             Special cases.

      IF (MWK(START(MB)+3) == 0 .OR. MWK(START(MC)+3) == 0 .OR.  &
          MWK(START(MA)+2) - 1 > MWK(START(MB)+2) + MWK(START(MC)+2) + NDIG) THEN
          RETURN
      ENDIF
      IF (MWK(START(MA)+3) == 0 .OR.  &
          MWK(START(MB)+2) + MWK(START(MC)+2) - 1 > MWK(START(MA)+2) + NDIG) THEN
          CALL FMMPY(MB,MC,MA)
          IF (MWK(START(MA)+2) /= MUNKNO) MWK(START(MA)) = -MWK(START(MA))
          RETURN
      ENDIF

      IF (ABS(MWK(START(MA)+2)) > MEXPAB .OR. ABS(MWK(START(MB)+2)) > MEXPAB .OR.       &
          ABS(MWK(START(MC)+2)) > MEXPAB .OR. MBASE < 1000 .OR. MBASE**2 > MAXINT .OR.  &
          NDIG > 900 .OR. NDIG > MAXINT/MBASE**2) THEN
          GO TO 110
      ENDIF

!             Determine which of abs(MA) and abs(MB*MC) is larger.

      MA_VS_MBMC = 0
      B = MBASE
      IF (MWK(START(MA)+2) <= MWK(START(MB)+2) + MWK(START(MC)+2) - 2) THEN
          MA_VS_MBMC = -1
          GO TO 120
      ENDIF
      IF (MWK(START(MA)+2) >= MWK(START(MB)+2) + MWK(START(MC)+2) + 1) THEN
          MA_VS_MBMC = 1
          GO TO 120
      ENDIF
      A1 = MWK(START(MA)+3)
      A2 = MWK(START(MA)+4)
      B1 = MWK(START(MB)+3)
      B2 = MWK(START(MB)+4)
      C1 = MWK(START(MC)+3)
      C2 = MWK(START(MC)+4)
      IF (MWK(START(MA)+2) == MWK(START(MB)+2) + MWK(START(MC)+2) - 1) THEN
          DPA = A1 * B + A2 + 1
          DPBC = B1 * C1 * B  +  B1 * C2  +  B2 * C1  +  C2 * B2 / B
          IF (DPA < DPBC) THEN
              MA_VS_MBMC = -1
              GO TO 120
          ENDIF

          DPA = A1 * B + A2
          DPBC = B1 * C1 * B  +  B1 * (C2+1)  +  (B2+1) * C1  +  (C2+1) * (B2+1) / B
          IF (DPA > DPBC) THEN
```

```
            MA_VS_MBMC = 1
            GO TO 120
         ENDIF
      ELSE IF (MWK(START(MA)+2) == MWK(START(MB)+2) + MWK(START(MC)+2)) THEN
         DPA = A1 * B + A2 + 1
         DPBC = B1 * C1  +  ( B1 * C2 + B2 * C1 ) / B  +  C2 * B2 / B**2
         IF (DPA < DPBC) THEN
            MA_VS_MBMC = -1
            GO TO 120
         ENDIF

         DPA = A1 * B + A2
         DPBC = B1 * C1  +  ( B1 * (C2+1) + (B2+1) * C1 ) / B  +  (C2+1) * (B2+1) / B**2
         IF (DPA > DPBC) THEN
            MA_VS_MBMC = 1
            GO TO 120
         ENDIF
      ENDIF

!            If it is not easy to determine which term is larger, make separate calls to
!            multiply and subtract.

  110 NUMBER_USED_SAVE = NUMBER_USED
      TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
      MXY = -2
      CALL FMMPY(MB,MC,MXY)
      CALL FMSUB_R1(MA,MXY)
      NUMBER_USED = NUMBER_USED_SAVE
      TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
      RETURN

!            Handle the operation using 4 cases, depending on which term is larger and whether
!            MA and MB*MC have opposite signs or not.

  120 IF (MWK(START(MA)) * MWK(START(MB)) * MWK(START(MC)) < 0) THEN
         IF (MA_VS_MBMC == 1) THEN
            MGD = 0
            K1 = 2 + MWK(START(MA)+2) - MWK(START(MB)+2) - MWK(START(MC)+2)
            DO J = K1, NDIG
               MBJ = MWK(START(MB)+3+J-K1)
               KPTA = START(MA)+2+J
               KPTC = START(MC)+3
               DO K = J, NDIG
                  MWK(KPTA) = MWK(KPTA) + MBJ * MWK(KPTC)
                  KPTA = KPTA + 1
                  KPTC = KPTC + 1
               ENDDO
               IF (KPTC <= START(MC)+2+NDIG) MGD = MGD + MBJ * MWK(KPTC)
            ENDDO
            K1 = MWK(START(MA)+2)
            MWK(START(MA)+2) = 0
            KPTA = START(MA)+3+NDIG
            MWK(KPTA-1) = MWK(KPTA-1) + NINT( MGD / MBASE )
            DO J = NDIG, 1, -1
               KPTA = KPTA - 1
               IF (MWK(KPTA) >= MBASE) THEN
```

```
            K = MWK(KPTA) / MBASE
            MWK(KPTA) = MWK(KPTA) - K * MBASE
            MWK(KPTA-1) = MWK(KPTA-1) + K
         ENDIF
      ENDDO
      IF (MWK(START(MA)+2) > 0) THEN
          DO J = NDIG, 1, -1
             MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
          ENDDO
          K1 = K1 + 1
      ENDIF
      MWK(START(MA)+2) = K1
      IF (MWK(START(MA)+3) >= MBASE) THEN
          DO J = NDIG, 3, -1
             MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
          ENDDO
          K = MWK(START(MA)+3) / MBASE
          MWK(START(MA)+4) = MWK(START(MA)+3) - K * MBASE
          MWK(START(MA)+3) = K
          MWK(START(MA)+2) = MWK(START(MA)+2) + 1
      ENDIF
   ENDIF

   IF (MA_VS_MBMC == -1) THEN
      MGD = 0
      K1 = MWK(START(MB)+2) + MWK(START(MC)+2) - 1
      K = MWK(START(MB)+2) + MWK(START(MC)+2) - MWK(START(MA)+2) - 1
      MWK(START(MA)+2) = 0
      IF (K > 0) THEN
          IF (K <= NDIG) MGD = MWK(START(MA)+2+NDIG+1-K)
          DO J = NDIG, K+1, -1
             MWK(START(MA)+2+J) = MWK(START(MA)+2+J-K)
          ENDDO
          DO J = 1, MIN(K,NDIG)
             MWK(START(MA)+2+J) = 0
          ENDDO
      ELSE IF (K == -1) THEN
          DO J = 1, NDIG
             MWK(START(MA)+1+J) = MWK(START(MA)+1+J+1)
          ENDDO
          MWK(START(MA)+2+NDIG) = 0
      ENDIF

      DO J = 1, NDIG
         MBJ = MWK(START(MB)+2+J)
         KPTA = START(MA)+2+J
         KPTC = START(MC)+3
         DO K = J, NDIG
            MWK(KPTA) = MWK(KPTA) + MBJ * MWK(KPTC)
            KPTA = KPTA + 1
            KPTC = KPTC + 1
         ENDDO
         IF (KPTC <= START(MC)+2+NDIG) MGD = MGD + MBJ * MWK(KPTC)
      ENDDO
      KPTA = START(MA)+3+NDIG
      MWK(KPTA-1) = MWK(KPTA-1) + NINT( MGD / MBASE )
```

```
        DO J = NDIG, 1, -1
           KPTA = KPTA - 1
           IF (MWK(KPTA) >= MBASE) THEN
               K = MWK(KPTA) / MBASE
               MWK(KPTA) = MWK(KPTA) - K * MBASE
               MWK(KPTA-1) = MWK(KPTA-1) + K
           ENDIF
        ENDDO
        IF (MWK(START(MA)+2) > 0) THEN
            DO J = NDIG, 1, -1
               MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
            ENDDO
            K1 = K1 + 1
        ENDIF
        MWK(START(MA)+2) = K1
        IF (MWK(START(MA)+3) >= MBASE) THEN
            DO J = NDIG, 3, -1
               MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
            ENDDO
            K = MWK(START(MA)+3) / MBASE
            MWK(START(MA)+4) = MWK(START(MA)+3) - K * MBASE
            MWK(START(MA)+3) = K
            MWK(START(MA)+2) = MWK(START(MA)+2) + 1
        ENDIF
    ENDIF

ELSE
    NUMBER_USED_SAVE = NUMBER_USED
    TEMPV_CALL_STACK = TEMPV_CALL_STACK + 1
    MXY = -2
    CALL FMEQ(MA,MXY)

    IF (MA_VS_MBMC == 1) THEN
        MGD = 0
        K1 = 2 + MWK(START(MA)+2) - MWK(START(MB)+2) - MWK(START(MC)+2)
        DO J = K1, NDIG
           MBJ = MWK(START(MB)+3+J-K1)
           KPTA = START(MA)+2+J
           KPTC = START(MC)+3
           DO K = J, NDIG
              MWK(KPTA) = MWK(KPTA) - MBJ * MWK(KPTC)
              KPTA = KPTA + 1
              KPTC = KPTC + 1
           ENDDO
           IF (KPTC <= START(MC)+2+NDIG) MGD = MGD - MBJ * MWK(KPTC)
        ENDDO
        K1 = MWK(START(MA)+2)
        MWK(START(MA)+2) = 0
        KPTA = START(MA)+3+NDIG
        MWK(KPTA-1) = MWK(KPTA-1) + NINT( MGD / MBASE )
        DO J = NDIG, 1, -1
           KPTA = KPTA - 1
           IF (MWK(KPTA) < 0) THEN
               K = (-MWK(KPTA)-1) / MBASE + 1
               MWK(KPTA) = MWK(KPTA) + K * MBASE
               MWK(KPTA-1) = MWK(KPTA-1) - K
```

```
      ELSE IF (MWK(KPTA) >= MBASE) THEN
          K = MWK(KPTA) / MBASE
          MWK(KPTA) = MWK(KPTA) - K * MBASE
          MWK(KPTA-1) = MWK(KPTA-1) + K
      ENDIF
    ENDDO
    IF (MWK(START(MA)+2) > 0) THEN
       DO J = NDIG, 1, -1
          MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
       ENDDO
       K1 = K1 + 1
    ENDIF
    MWK(START(MA)+2) = K1
    IF (MWK(START(MA)+3) == 0) THEN
       CALL FMMPY(MB,MC,MA)
       CALL FMSUB_R2(MXY,MA)
       NUMBER_USED = NUMBER_USED_SAVE
       TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
       RETURN
    ENDIF
ENDIF

IF (MA_VS_MBMC == -1) THEN
    MGD = 0
    K1 = MWK(START(MB)+2) + MWK(START(MC)+2) - 1
    K = MWK(START(MB)+2) + MWK(START(MC)+2) - MWK(START(MA)+2) - 1
    MWK(START(MA)+2) = 0
    IF (K > 0) THEN
       IF (K <= NDIG) MGD = -MWK(START(MA)+2+NDIG+1-K)
       DO J = NDIG, K+1, -1
          MWK(START(MA)+2+J) = -MWK(START(MA)+2+J-K)
       ENDDO
       DO J = 1, MIN(K,NDIG)
          MWK(START(MA)+2+J) = 0
       ENDDO
    ELSE IF (K == -1) THEN
       DO J = 1, NDIG
          MWK(START(MA)+1+J) = -MWK(START(MA)+1+J+1)
       ENDDO
       MWK(START(MA)+2+NDIG) = 0
    ELSE IF (K == 0) THEN
       DO J = 1, NDIG
          MWK(START(MA)+2+J) = -MWK(START(MA)+2+J)
       ENDDO
    ENDIF

    DO J = 1, NDIG
       MBJ = MWK(START(MB)+2+J)
       KPTA = START(MA)+2+J
       KPTC = START(MC)+3
       DO K = J, NDIG
          MWK(KPTA) = MWK(KPTA) + MBJ * MWK(KPTC)
          KPTA = KPTA + 1
          KPTC = KPTC + 1
       ENDDO
       IF (KPTC <= START(MC)+2+NDIG) MGD = MGD + MBJ * MWK(KPTC)
```

```
        ENDDO
        KPTA = START(MA)+3+NDIG
        MWK(KPTA-1) = MWK(KPTA-1) + NINT( MGD / MBASE )
        DO J = NDIG, 1, -1
           KPTA = KPTA - 1
           IF (MWK(KPTA) >= MBASE) THEN
               K = MWK(KPTA) / MBASE
               MWK(KPTA) = MWK(KPTA) - K * MBASE
               MWK(KPTA-1) = MWK(KPTA-1) + K
           ELSE IF (MWK(KPTA) < 0) THEN
               K = (-MWK(KPTA)-1) / MBASE + 1
               MWK(KPTA) = MWK(KPTA) + K * MBASE
               MWK(KPTA-1) = MWK(KPTA-1) - K
           ENDIF
        ENDDO
        IF (MWK(START(MA)+2) > 0) THEN
            DO J = NDIG, 1, -1
               MWK(START(MA)+2+J) = MWK(START(MA)+2+J-1)
            ENDDO
            K1 = K1 + 1
        ENDIF
        MWK(START(MA)+2) = K1
        MWK(START(MA)) = -MWK(START(MA))
        IF (MWK(START(MA)+3) == 0) THEN
            CALL FMMPY(MB,MC,MA)
            CALL FMSUB_R2(MXY,MA)
            NUMBER_USED = NUMBER_USED_SAVE
            TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
            RETURN
        ENDIF
    ENDIF

    NUMBER_USED = NUMBER_USED_SAVE
    TEMPV_CALL_STACK = TEMPV_CALL_STACK - 1
    RETURN
ENDIF

END SUBROUTINE FMMPY_SUB
```