

```
FUNCTION FM_FPRIME(N,A,F,NF)
USE FMVALS
USE FMZM
IMPLICIT NONE
```

```
! This routine finds the Nth derivative of F(X,NF), evaluated at A.
! NF is passed on to function F to indicate which function to use in cases where several
! different functions may be defined there.

! F must be defined in an interval containing A, so that F can be sampled on both sides of A.

! N may be zero, so that in cases where F suffers cancellation error at A, an accurate
! function value is returned.

! FM_FPRIME tries to return full accuracy for the derivative, by raising precision above
! the user's level and using difference formulas.
```

```
TYPE (FM)          :: FM_FPRIME, A
TYPE (FM), EXTERNAL :: F
INTEGER :: J, K, KWARN_SAVE, NDSAVE, N, NF
TYPE (FM), SAVE :: D1, D2, ERR, F1, F2, H, TOL, TOL2, X1
```

```
CALL FM_ENTER_USER_FUNCTION(FM_FPRIME)
```

```
!           Raise precision slightly.
```

```
NDSAVE = NDIG
NDIG = NDIG + NGRD52
CALL FM_EQU(A,X1,NDSAVE,NDIG)
KWARN_SAVE = KWARN
KWARN = 0
```

```
ERR = 1
D2 = 0
F1 = F(X1,NF)
IF (F1 /= 0) THEN
  CALL FM_ULP(F1,TOL)
ELSE
  TOL = EPSILON(TO_FM(1))
ENDIF
TOL = ABS(TOL)
```

```
!           Check for a legal function value.
```

```
IF (IS_UNKNOWN(F1) .OR. IS_OVERFLOW(F1) .OR. IS_UNDERFLOW(F1) .OR. N < 0) THEN
  FM_FPRIME = TO_FM(' UNKNOWN ')
  GO TO 110
ENDIF
F2 = F1
```

```
!           Loop at increasing precision until the difference formula is accurate.
```

```
DO J = 1, 100

  NDIG = 2*NDIG
```

! Define the variables used below at the new higher precision.

```
CALL FM_EQU(D2,D1,NDIG/2,NDIG)
CALL FM_EQU(F2,F1,NDIG/2,NDIG)
CALL FM_EQU(TOL,TOL2,NDSAVE,NDIG)
CALL FM_EQU(A,X1,NDSAVE,NDIG)
```

! Special case for N = 0.

```
IF (N == 0) THEN
  F2 = F(X1,NF)
  D2 = F2
  IF (ABS(F2-F1) < TOL2) GO TO 110
  CYCLE
ENDIF
F2 = F1
```

! Special case for N = 1.

```
IF (N == 1) THEN
  IF (X1 /= 0) THEN
    CALL FM_ULP(X1,H)
  ELSE
    H = EPSILON(TO_FM(1))
  ENDIF
  H = SQRT(ABS(H))
  D2 = ( F(X1+H,NF) - F(X1-H,NF) ) / (2*H)
  IF (ABS(D2-D1) < TOL2 .AND. J > 1) GO TO 110
  CYCLE
ENDIF
```

! General case for even N > 1.

```
IF (MOD(N,2) == 0) THEN
  IF (X1 /= 0) THEN
    CALL FM_ULP(X1,H)
  ELSE
    H = EPSILON(TO_FM(1))
  ENDIF
  H = ABS(H)**(TO_FM(1)/(N+2))
  D2 = (-1)**(N/2) * BINOMIAL(TO_FM(N),TO_FM(N/2)) * F(X1,NF)
  DO K = 0, N/2-1
    D2 = D2 + (-1)**K * BINOMIAL(TO_FM(N),TO_FM(K)) * &
      ( F(X1+(N/2-K)*H,NF) + F(X1-(N/2-K)*H,NF) )
  ENDDO
  D2 = D2 / H**N
  IF (ABS(D2-D1) < TOL2 .AND. J > 1) GO TO 110
  CYCLE
ENDIF
```

! General case for odd N > 1.

```
IF (MOD(N,2) == 1) THEN
  IF (X1 /= 0) THEN
    CALL FM_ULP(X1,H)
```

```

ELSE
    H = EPSILON(TO_FM(1))
ENDIF
H = ABS(H)**(TO_FM(1)/(N+2))
D2 = 0
DO K = 0, N/2
    D2 = D2 + (-1)**K * BINOMIAL(TO_FM(N-1),TO_FM(K)) * &
        ( F(X1+(N/2-K+1)*H,NF) - F(X1-(N/2-K+1)*H,NF) ) * &
        TO_FM(N*(N+1-2*K)) / ((N-K)*(N+1-K))
ENDDO
D2 = D2 / (2**H**N)
IF (ABS(D2-D1) < TOL2 .AND. J > 1) GO TO 110
CYCLE
ENDIF

```

```
ENDDO
```

```
!           Round and return.
```

```

110 CALL FM_EQU(D2,FM_FPRIME,NDIG,NDSAVE)
    NDIG = NDSAVE
    KWARN = KWARN_SAVE
    CALL FM_EXIT_USER_FUNCTION(FM_FPRIME)
    END FUNCTION FM_FPRIME

```