NAG Library Routine Document

X06ADF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

X06ADF returns the OpenMP thread number of the calling thread.

2 Specification

```
FUNCTION X06ADF ()
INTEGER X06ADF
```

3 Description

X06ADF, for multi-threaded implementations, returns the calling OpenMP thread's unique thread number within the current team. The master thread will always return 0. The remaining threads will return a value between 1 and the value returned by X06ABF less 1.

If this routine is called from a sequential part of a multi-threaded program then it will return the value 0.

In serial implementations of the NAG Library this routine will always return 0. See the X06 Chapter Introduction for a discussion of the behaviour of these routines when called in serial.

4 References

OpenMP Specifications http://openmp.org/wp/OpenMP-Specifications

Chapman B, Jost G and van der Pas R (2008) *Using OpenMP Portable Shared Memory Parallel Programming* The MIT Press

5 Arguments

None.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

X06ADF is not threaded in any implementation.

9 Further Comments

None.

10 Example

In this example we presume a multi-threaded implementation of the NAG Library. We call X06ADF both outside and inside an OpenMP active parallel region. Outside we expect a single thread to display the value 0. Inside the region we use the value to have only the master thread display the result.

We also call X06AFF inside and outside of the region. Outside we expect it to return 0, as we are not in an active parallel region, and inside we expect to see the value 1, indicating that the parallel region is an active one.

If you use a serial implementation of the NAG library, regardless of whether the code is compiled with OpenMP or not, X06ABF will always return 1 and X06ADF and X06AFF will always return 0. The appropriate results file will be included with the distribution material for your implementation.

10.1 Program Text

```
    Program x06adfe

!     X06ADF Example Program Text

!     Mark 26 Release. NAG Copyright 2015.

!     .. Use Statements ..
      Use nag_library, Only: x06abf, x06adf, x06aff
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter               :: nout = 6
!     .. Local Scalars ..
      Integer                          :: in_para, me, num_got, num_req
!     .. Executable Statements ..
      Write (nout,*) 'X06ADF Example Program Results'
      Write (nout,*)

!     Get the thread number with a call to X06ADF and display it
      me = x06adf()
      Write (nout,99999) 'Thread number:            ', me
      Write (nout,*)

!     Call x06aff to check whether we are in an active parallel region
      in_para = x06aff()
      Write (nout,99999) 'In active parallel region:', in_para
      Write (nout,*)

!     Spawn an OpenMP parallel region, have the master thread display
!     the number of threads and check whether we are in an active
!     parallel region

      num_req = 5

      !$Omp Parallel Num_threads (num_req), Private (in_para,me,num_got),      &
      !$Omp   Default (None)

        me = x06adf()
        num_got = x06abf()
        in_para = x06aff()

        If (me==0) Then
          Write (nout,99999) 'Number of threads:        ', num_got
          Write (nout,*)
          Write (nout,99999) 'In active parallel region:', in_para
          Write (nout,*)
        End If

      !$Omp End Parallel

99999 Format (1X,A,I11)

    End Program x06adfe
```

10.2 Program Data

None.

10.3 Program Results

```
 X06ADF Example Program Results

 Thread number:                    0

 In active parallel region:        0

 Number of threads:                5

 In active parallel region:        1
```