HSL_ZD11

1 SUMMARY

This package defines a derived type capable of supporting a variety of sparse matrix storage schemes. Its principal use is to allow exchange of data between HSL subprograms and other codes.

ATTRIBUTES — **Version:** 1.1.0. (6 March 2007) **Types:** Real (single, double), Integer, Complex (single, double). **Remark:** This package is also included in the HSL Archive. **Original date:** February 2006. **Origin:** N. I. M. Gould and J. K. Reid, Rutherford Appleton Laboratory. **Language:** Fortran 95 + TR 15581 (allocatable components). **Licence:** A third-party licence for this package is available without charge.

2 HOW TO USE THE PACKAGE

Access to the package requires a `USE` statement such as

Single precision version

```
USE HSL_ZD11_single
```

Double precision version

```
USE HSL_ZD11_double
```

Integer version

```
USE HSL_ZD11_integer
```

Complex version

```
USE HSL_ZD11_complex
```

Double complex version

```
USE HSL_ZD11_double_complex
```

If it is required to use more than one of these modules at the same time, the derived type `ZD11_type` (Section 2.1) must be renamed and the procedures `ZD11_put` and `ZD11_get` (Section 2.2) must be accessed only once, for example,

```
USE HSL_ZD11_double
USE HSL_ZD11_double_complex, only : ZD11_complex_type => ZD11_type
```

2.1 The derived data type

A single derived data type, `ZD11_type`, is accessible from the package. It is intended that, for any particular application, only those components which are needed will be set. The components are:

`id` is a allocatable array of rank one and type default `CHARACTER(1)` that may be used to identify the matrix.

`type` is a allocatable array of rank one and type default `CHARACTER(1)` that may be used to indicate the properties of the matrix in question.

`m` is a scalar component of type default `INTEGER` that may be used to hold the number of rows in the matrix.

`n` is a scalar component of type default `INTEGER` that may be used to hold the number of columns in the matrix.

`ne` is a scalar component of type default `INTEGER` that may be used to hold the number of entries in the matrix.

`row` is a allocatable array of rank one and type default `INTEGER` that may be used to hold the row indices of the entries of the matrix.

`col` is a allocatable array of rank one and type default `INTEGER` that may be used to hold the column indices of the entries of the matrix.

val is a allocatable array of rank one and type default REAL (double precision REAL in HSL_ZD11_double, COMPLEX in HSL_ZD11_complex, COMPLEX(KIND(0.0D0) in HSL_ZD11_double_complex, INTEGER in HSL_ZD11_integer) that may be used to hold the numerical values of the entries of the matrix.

ptr is a allocatable array of rank one and type default INTEGER that may be used to hold the starting positions of each row in a row-wise storage scheme, or the starting positions of each column in a column-wise storage scheme.

2.2 Conversions between character variables and character arrays

To assist use of the character arrays in the components id and type, the module provides two procedures:

ZD11_put is a subroutine that allocates a character array and sets its components from a character variable.

ZD11_get is a function that obtains the elements of a character array as a character variable.

Allocate a character array and set its components

```
CALL ZD11_put(array,string,stat)
```

array is a rank-one allocatable array of type CHARACTER(LEN=1). If it is allocated on entry, it is deallocated; if the deallocation is unsuccessful, there is an immediate return with stat having a nonzero value. Next, array is allocated with size LEN_TRIM(string); if the allocation is unsuccessful, there is an immediate return with stat having a nonzero value; otherwise, the elements of array are given the values string(i:i), i = 1, 2, ... LEN_TRIM(string).

string is a scalar of INTENT(IN) and type CHARACTER with any character length.

stat is a scalar of INTENT(OUT) and type default INTEGER. The DEALLOCATE and ALLOCATE statements are given this as their STAT= variable and a successful execution will be indicated by the value zero.

Obtain the elements of a character array as a character variable

```
string = ZD11_get(array)
```

array is a rank-one array of INTENT(IN) and type CHARACTER(LEN=1).

The result is scalar and of type CHARACTER(LEN=SIZE(array)). ZD11_get(i:i) is given the value array(i), i = 1, 2, ..., SIZE(array).

3 GENERAL INFORMATION

Other modules used directly: None.

Input/output: None.

Restrictions: None.

5 EXAMPLE OF USE

The following code stores the upper-triangular part of the symmetric matrix

$$\begin{pmatrix} 1.0 & & \\ & & 1.0 \\ & 1.0 & \end{pmatrix},$$

whose identifier is “Sparse”, using a coordinate sparse matrix storage format. It writes out details of the stored data and then deallocates the allocatable components.

```
PROGRAM MAIN
  USE HSL_ZD11_double
   INTEGER :: i,stat
   TYPE ( ZD11_type ) :: A
   A%n = 3 ; A%ne = 2

   ALLOCATE( A%row( A%ne ), A%col( A%ne ), A%val( A%ne ) )
   CALL ZD11_put  (A%id,'Sparse',stat)
   A%row( 1 ) = 1 ; A%col( 1 ) = 1 ; A%val( 1 ) = 1.0
   A%row( 2 ) = 2 ; A%col( 2 ) = 3 ; A%val( 2 ) = 1.0

   WRITE( 6, "( 3A, I2, //, A )" ) ' Matrix ', ZD11_get(A%id), &
          ' dimension', A%n, ' row col  value '
   DO i = 1, A%ne
      WRITE( 6, "( I3, 1X, I3, ES9.1 )" ) A%row( i ), A%col( i ), A%val( i )
   END DO
   DEALLOCATE( A%id, A%row, A%col, A%val)
END PROGRAM MAIN
```

This produces the following output:

```
 Matrix Sparse dimension 3

 row col  value
  1   1  1.0E+00
  2   3  1.0E+00
```