Chapter 19

Library Utility Subprograms
19.1 System Parameters

A. Purpose

These subprograms provide values of various system parameters that are needed in library
subprograms and typically have different values on different computer systems.

B. Usage

Program Prototype

REAL R1MACH, S
DOUBLE PRECISION D1MACH, D
INTEGER I1MACH, I, J

Set J in the range 1 ≤ J ≤ 5 for R1MACH or D1MACH or 1 ≤ J ≤ 16 for I1MACH. Then, use the
appropriate one of the following statements:

\[ S = R1MACH(J) \]
\[ D = D1MACH(J) \]
\[ I = I1MACH(J) \]

The results, S, D, or I are set as described in Section D.

Argument Definitions

J  [in] Integer argument selecting the desired system parameter as described in Section D.

C. Examples and Remarks

The program DRMACH lists all the values obtainable from R1MACH, D1MACH, and
I1MACH on the host computer system. Output is shown from several different host systems.

D. Functional Description

For the purpose of this package, a model of the Fortran 77 INTEGER, REAL, and DOUBLE
PRECISION number sets is characterized by a total of nine fundamental parameters.

The model of Fortran 77 numbers of type INTEGER is parameterized by two numbers, \( a \) and \( s \),
where \( a \) denotes the base (radix) of the number system and \( s \) denotes the maximum number of
base-\( a \) digits available to represent a Fortran integer. Thus the integers range from \(-a^s-1\) to \(a^s-1\).

The model of Fortran 77 numbers of type REAL is characterized by four parameters, \( b \), \( t \), \( emin \),
and \( emax \); where \( b \) is the base of the fraction part, \( t \) is the number of base-\( b \) digits in the fraction
part, \( emin \) is the minimum exponent and \( emax \) is the maximum exponent. The magnitude of a
floating-point number is thus of the form

\[ b^e (c_1 b^{-1} + c_2 b^{-2} + \ldots + c_t b^{-t}) \]

where \( emin \leq e \leq emax \), and the digits \( c_i \) satisfy \( 0 \leq c_i \leq b-1 \).

A nonzero floating-point number is normalized if and only if the
digit \( c_1 \) is nonzero. We shall consider only normalized floating-point numbers, although numeric
processors based on the IEEE standard, such as the Intel 8087 family, support a range of
unnormalized numbers.

Fortran 77 numbers of type DOUBLE PRECISION are modeled in the same form as REAL
numbers and are assumed to have the same base, \( b \), but generally different values of \( t \), \( emin \), and
\( emax \).
For some computer systems no setting of these parameters will make the model system coincide exactly with the actual computer’s number set. In such cases the model parameters are selected so the model system will be as large a subset of the actual number set as possible. In particular, for Cray systems the parameters $t$ and $e_{max}$ are set smaller and $e_{min}$ is set larger than one might expect from the structure of floating-point numbers on Cray systems. The reasons for this are described in Ref 3.

The values returned by this package are either prestored constants or are computed at compile-time from prestored constants by use of expressions in parameter statements. Thus correct values must be determined and edited into the package whenever this package is moved to a new computer system. Correct values for many systems are present as comments in the source code.

**Specification of Values Returned**

$I1MACH (J)$

- 1. Standard input unit number.
- 2. Standard output unit number.
- 4. Standard error message unit number.
- 5. Number of bits per Fortran integer storage unit.
- 6. Number of characters per Fortran integer storage unit.
- 7. $a$, the base for integers.
- 8. $s$, the number of base-$a$ digits in an integer.
- 9. $a^{s-1}$, the largest integer magnitude.
- 10. $b$, the base for floating-point numbers. Assumed the same for REAL and DOUBLE PRECISION arithmetic.
- 11. $t$, the number of base-$b$ digits for REAL arithmetic.
- 12. $e_{min}$, minimum exponent for REAL arithmetic.
- 13. $e_{max}$, maximum exponent for REAL arithmetic.
- 14. $t$, the number of base-$b$ digits for DOUBLE PRECISION arithmetic.
- 15. $e_{min}$, minimum exponent for DOUBLE PRECISION arithmetic.
- 16. $e_{max}$, maximum exponent for DOUBLE PRECISION arithmetic.

$d_{1MACH}(J)$

- 1. $b^{e_{min}-1}$, smallest positive normalized REAL number, [underflow limit].
- 2. $b^{e_{max}(1-b^{-t})}$, largest REAL number, [overflow limit].
- 3. $b^t$, smallest relative difference between two successive nonzero REAL numbers. This is also the difference between 1.0 and the next smaller REAL number.
- 4. $b^{t+1}$, largest relative difference between two successive nonzero REAL numbers. This is also the difference between 1.0 and the next larger REAL number.
- 5. log$_{10}b$, useful in certain conversions between base $b$ and base 10.

The values returned by D1MACH are as described above for R1MACH with REAL replaced by DOUBLE PRECISION.

**Historical Perspective and Relations to Other Languages**

The specifications of R1MACH, D1MACH, and I1MACH and the original implementation were developed at the AT&T Bell Laboratories, Murray Hill, New Jersey, in the 70’s to support the development of portable mathematical software, and specifically the PORT library (Ref. 1) which is a proprietary AT&T Bell Laboratories product. These three subprograms were published as a subset of Algorithm 528 in TOMS (Ref. 2) and are not proprietary. The present implementation for MATH77 is substantially different from the original versions.
The attributes associated with J = 1, 2, 3, 4, and 6 in I1MACH are less relevant in Fortran 77 in the 90’s than they were in Fortran 66 in the 70’s. In particular, other MATH77 library subprograms do not call I1MACH to obtain the Unit number for printing. Instead printing uses PRINT or WRITE(*,...).

Languages developed more recently than Fortran 77, such as Ada, ANSI C, and Fortran 90, provide methods within the language to obtain certain environmental parameters. Consider, for example, the underflow and overflow limits, and precision for floating-point arithmetic. Using the present package, these can be obtained for DOUBLE PRECISION arithmetic by referencing D1MACH(1), D1MACH(2), and D1MACH(4), respectively, and for REAL arithmetic by referencing R1MACH(1), R1MACH(2), and R1MACH(4). In Fortran 90 these parameters can be obtained by referencing the generic inquiry functions TINY(X), HUGE(X), and EPSILON(X); where X may be any DOUBLE PRECISION entity to obtain the values for DOUBLE PRECISION arithmetic, and any REAL entity to obtain the values for REAL arithmetic. In ANSI C these values for arithmetic of type long double are given by the macro names LDBL_MIN, LDBL_MAX, and LDBL_EPSILON, which are defined in the standard header file float.h. Similarly, for type double there are DBL_MIN, DBL_MAX, and DBL_EPSILON, and for type float there are FLT_MIN, FLT_MAX, and FLT_EPSILON.

**References**


**E. Error Procedures and Restrictions**

If the argument is outside the range 1 ≤ J ≤ 16 for I1MACH or outside 1 ≤ J ≤ 5 for R1MACH or D1MACH, an error message is printed and execution is terminated.

This package contains a partial protection against the inadvertent use of the wrong version of one of these subroutines; say using the PC version on a VAX. On the first call to any one of these subprograms, tests are done to verify that two of the stored parameter values are not grossly wrong for the current environment. These tests depend on assumptions about hardware, compilers, and linkers that may be invalidated by technological changes. Subroutines AMTEST and AMSUB1 are used to support these tests and are not intended for any other usage.

**F. Supporting Information**

The source language is ANSI Fortran 77. All six of these program units are grouped into a single file, AMACH.FOR. The extension ".FOR" will be different on different systems, e.g., ".f" on UNIX systems. This grouping makes it easier to inspect and edit D1MACH, R1MACH, and I1MACH in a consistent way when installing the MATH77 library for a new computer or compiler.

<table>
<thead>
<tr>
<th>Program Unit and Entry Names</th>
<th>External References</th>
</tr>
</thead>
<tbody>
<tr>
<td>I1MACH</td>
<td>AMACH</td>
</tr>
<tr>
<td>D1MACH</td>
<td>AMACH</td>
</tr>
<tr>
<td>R1MACH</td>
<td>AMACH</td>
</tr>
<tr>
<td>AMACH</td>
<td>AMACH</td>
</tr>
</tbody>
</table>

System Parameters 19.1-3
DRMACH

program DRMACH

      Demo driver for D1MACH, I1MACH, and R1MACH

      DOUBLE PRECISION D1MACH, D1
      INTEGER I1MACH, I1
      REAL R1MACH, R1

      !---------------------------------------------------------
      PRINT 100,'MACHINE CONSTANTS for ...                
      PRINT 200
      PRINT 300
      PRINT*, ' 
      DO J=1,5            ! Using Fortran 90 "DO" syntax
        I1 = I1MACH(J)
        R1 = R1MACH(J)
        D1 = D1MACH(J)
        PRINT 400, J,I1,R1,D1
      ENDDO ! J
      DO J=6,16
        PRINT 500, J,I1MACH(J)
      ENDDO ! J

      !---------------------------------------------------------
      100  FORMAT(' ',14X,A)
      200  FORMAT(' ',14X,41('-'))
      300  FORMAT('0',1X,'J',6X,'I1MACH(J)',7X,'R1MACH(J)',13X,'D1MACH(J)')
      400  FORMAT(' ',I2, 3X,I12,3X,G15.8,3X,G25.18)
      500  FORMAT(' ',I2,3X,I12)
      END

      MACHINE CONSTANTS for IEEE Arithmetic
      -------------------------------------

<table>
<thead>
<tr>
<th>J</th>
<th>I1MACH(J)</th>
<th>R1MACH(J)</th>
<th>D1MACH(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.11754940E-37</td>
<td>0.222507385850720100-307</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.34028230E+39</td>
<td>0.179769313486231600+309</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.59604640E-07</td>
<td>0.111022302462515600E-15</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.11920930E-06</td>
<td>0.222044604925031300E-15</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.30103000</td>
<td>0.30102995663981200</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2147483647</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-125</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-1021</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1024</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
MACHINE CONSTANTS for VAX
-------------------------

<table>
<thead>
<tr>
<th>J</th>
<th>I1MACH(J)</th>
<th>R1MACH(J)</th>
<th>D1MACH(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.29387359E-38</td>
<td>0.29387358770571877E-38</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.17014117E+39</td>
<td>0.170141183460469229E+39</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>0.59604645E-07</td>
<td>0.13877878078144568E-16</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.11920929E-06</td>
<td>0.277555756156289135E-16</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>0.30103001</td>
<td>0.301029995663981198</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2147483647</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>127</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MACHINE CONSTANTS for CRAY Y/MP
-------------------------------

<table>
<thead>
<tr>
<th>J</th>
<th>I1MACH(J)</th>
<th>R1MACH(J)</th>
<th>D1MACH(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>0.73344155-2465</td>
<td>0.733441547021938866-2465</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.13634352+2466</td>
<td>0.136343516952426991+2466</td>
</tr>
<tr>
<td>3</td>
<td>102</td>
<td>0.71054274E-14</td>
<td>0.50487097934144755E-28</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>0.14210855E-13</td>
<td>0.100974195868289511E-27</td>
</tr>
<tr>
<td>5</td>
<td>64</td>
<td>0.30103000</td>
<td>0.301029995663981195</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>70368744177663</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-8188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>8189</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-8188</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8189</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MACHINE CONSTANTS for UNISYS 1100
---------------------------------

<table>
<thead>
<tr>
<th>J</th>
<th>I1MACH(J)</th>
<th>R1MACH(J)</th>
<th>D1MACH(J)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>.14693679-038</td>
<td>.278134232313400172-308</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>.17014118+039</td>
<td>.898846567431157951+308</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>.74505806-008</td>
<td>.867361737988403547-018</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>.14901161-007</td>
<td>.173472347597680709-017</td>
</tr>
<tr>
<td>5</td>
<td>36</td>
<td>.30103000</td>
<td>.301029995663981194</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>34359738367</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>27</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>-128</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>127</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>-1024</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>1023</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
19.2  Error Message Processor

A. Purpose

This package of subroutines processes diagnostic messages, providing options to write or not write the message, and to return or stop after message processing. These subroutines are intended primarily for use by other library subprograms although they are not restricted to that usage.

By passing error messages through these subroutines it is possible to (a) avoid having Fortran I/O statements in other library subprograms that do not otherwise execute any I/O functions, and (b) provide a means, via the subroutine ERMSET, for a user program to alter the nominal action of error message processing.

B. Usage

To process an error message, the program detecting the error must make a sequence of one or more calls to subroutines of this package. Such a sequence of calls is distinguished by the condition that the argument, FLAG, has the value ‘,’ in all but the last call, and has the value ‘.’ in the last call of the sequence.

The first, and possibly only, call of a sequence must be to ERMSG, SERM1, DERM1, IERM1, SERMN, or DERMN. These subroutines all have the argument, LEVEL, that specifies the nominal action level for the entire sequence of calls.

If there are additional calls in the sequence, the subsequent calls must be to SERV1, DERV1, IERV1, SERVN, DERVN, or ERMOR, each of which may be called any number of times. These calls provide additional data values or character strings to be included in the printed error message.

The package contains two additional subroutines: ERMSET, which can be called by a user to alter the nominal action of the package, and ERFIN which is called by other subroutines of the package when FLAG = ‘.’ to handle the common final steps of processing an error message.

Type Statements for Arguments

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Restrictions</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>IERR, LEVEL, IDATA, IDELTA, NVAL</td>
<td></td>
</tr>
<tr>
<td>CHARACTER</td>
<td>SUBNAM*{n}, MESS*{n}_2, FLAG, LABEL*{n}_3</td>
<td>[n, n_2, n_3 &gt; 0]</td>
</tr>
<tr>
<td>CHARACTER*{n}_4</td>
<td>LABELS(≥NVAL)</td>
<td>[n_4 ≥ 0]</td>
</tr>
<tr>
<td>REAL</td>
<td>SDATA, SVALS(≥NVAL)</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>DDATA, DVALS(≥NVAL)</td>
<td></td>
</tr>
</tbody>
</table>

Call Statements

In each of these calls, all arguments must be assigned values before the call and none of the argument values will be changed by the subroutine. It will usually be most convenient to supply most of the arguments, particularly those of type character, as literals in the call statement.

To initiate an error message:

```
CALL ERMSG (SUBNAM, IERR, LEVEL, MESS, FLAG)
```

To initiate a message, including one REAL data item:

```
CALL SERM1 (SUBNAM, IERR, LEVEL, MESS, LABEL, SDATA, FLAG)
```
To initiate a message, including one DOUBLE PRECISION data item:

```
CALL DERM1 (SUBNAM, IERR, LEVEL, MESS, LABEL, DDATA, FLAG)
```

To initiate a message, including one INTEGER data item:

```
CALL IERM1 (SUBNAM, IERR, LEVEL, MESS, LABEL, IDATA, FLAG)
```

To initiate a message, including an array of REAL data:

```
CALL SERMN (SUBNAM, IERR, LEVEL, MESS, NVAL, LABELS, SVALS, FLAG)
```

To initiate a message, including an array of DOUBLE PRECISION data:

```
CALL DERMN (SUBNAM, IERR, LEVEL, MESS, NVAL, LABELS, DVALS, FLAG)
```

To add a REAL datum to the current message:

```
CALL SERV1 (LABEL, SDATA, FLAG)
```

To add a DOUBLE PRECISION datum to the current message:

```
CALL DERV1 (LABEL, DDATA, FLAG)
```

To add an INTEGER datum to the current message:

```
CALL IERV1 (LABEL, IDATA, FLAG)
```

To add an array of REAL data to the current message:

```
CALL SERVN (NVALS, LABELS, SVALS, FLAG)
```

To add an array of DOUBLE PRECISION data to the current message:

```
CALL DERNV (NVALS, LABELS, DVALS, FLAG)
```

To add an additional character string, MESS, to the current message:
CALL ERMOR (MESS, FLAG)

To alter the nominal message processing action:

CALL ERMSET (IDELTA)

Subroutine called by other subroutines of the package when FLAG = '.' to print the closing line of dollar signs and take the final action of returning or stopping:

CALL ERFIN

**Argument Definitions**

**SUBNAM**[in] Name of subprogram in which error has been detected. Suggest length of name ≤ 12.

**IERR**[in] Identification number for the error.

**LEVEL**[in] Should be set to 2, 0, or -2 to specify the nominal action desired.

  2 = print and stop
  0 = print and return
  -2 = return with no printing

  This specification applies over a complete sequence of calls to the package as described above. In particular if a stop is to occur, it will not happen until the last call of a sequence, i.e. a call with FLAG = '.'.

  The actual action taken is governed by: ALPHA = NDELTA + LEVEL,

  where NDELTA is a saved local variable in the package. The initial value of NDELTA is zero but it can be changed by use of subroutine ERMSET.

**MESS**[in] Error message of length ≤ 72.

**FLAG**[in] A single character, either a comma or a period. A comma means further data or character strings to be included in the error message will be provided by subsequent calls. A period means this is the last call relating to the current error message.

**LABEL**[in] An identifying label to be printed with the data value given in SDATA, DDATA, or IDATA. Suggest length of label ≤ 35.

**SDATA**[in] Data item of type REAL to be printed with the error message.

**DDATA**[in] Data item of type DOUBLE PRECISION to be printed with the error message.

**IDATA**[in] Data item of type INTEGER to be printed with the error message.

**NVAL**[in] Number of array elements to be printed from LABELS() and from SVALS() or DVALS().

**LABELS**()[in] An array of NVAL labels to be printed with the data values given in SVALS() or DVALS(). The array elements should have CHARACTER length ≤ 35.
SVALS() [in] An array of NVAL REAL values to be printed.
DVALS() [in] An array of NVAL DOUBLE PRECISION values to be printed.
IDELTA [in] New value to be assigned to the saved local variable NDELTA. Alters the action of the error processing.

C. Examples and Remarks
Usage is illustrated by the program DRERMSG and the output ODERMSG.

D. Functional Description
Levels of Action
The actual action level, NALPHA, computed as LEVEL + NDELTA, determines the action as follows:

- NALPHA ≥ 2 print message and STOP.
- NALPHA = -1, 0 or 1 print message and RETURN.
- NALPHA ≤ -2 RETURN, doing no printing.

Effects of Resetting NDELTA
The saved local variable, NDELTA, initially has the value zero. If it is changed by a call to ERMSSET the effect can be interpreted as follows:

<table>
<thead>
<tr>
<th>NDELTA</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Print and stop on all diagnostics that would otherwise be printed.</td>
</tr>
<tr>
<td>0</td>
<td>Take the standard action.</td>
</tr>
<tr>
<td>-1</td>
<td>Do not stop on any diagnostics. Print as usual.</td>
</tr>
<tr>
<td>-2</td>
<td>Never stop. Print only those diagnostics that nominally result in a stop.</td>
</tr>
<tr>
<td>-4</td>
<td>Do not print or stop on any diagnostic.</td>
</tr>
</tbody>
</table>

Form of the Error Message
The message will begin with a line of 72 dollar signs. The next two lines will be:

Subprogram SUBNAM reports Error No. IERR
The initial message, MESS.

Following may be lines of the following forms:

1. `LABEL = value` where value is SDATA, DDATA, or IDATA,
2. `label1 = val1, label2 = val2, ...` where the labels are from LABELS() and the values are from SVALS() or DVALS(), or
3. `MESS` transmitted by subroutine ERMSOR.

Finally the message will be terminated with another line of 72 dollar signs.

E. Error Procedures and Restrictions
This package shares data via a COMMON block named M77ERR. The user must not use this name for any other COMMON block.

If any of the character string arguments are longer than the suggested maximum lengths, the corresponding printed line of the error message will exceed a length of 72 characters.

The subroutine ERMSSET should not be called between the beginning and end of a sequence of calls to the error processing subroutines.
F. Supporting Information

The source language is ANSI Fortran 77. Uses common block M77ERR.

<table>
<thead>
<tr>
<th>Program Unit</th>
<th>Entry Names</th>
<th>External Refs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERMST</td>
<td>ERMSG, ERMSET</td>
<td>ERFIN</td>
</tr>
<tr>
<td>ERMOR</td>
<td>ERMOR</td>
<td>ERFIN</td>
</tr>
<tr>
<td>SERM1</td>
<td>SERM1</td>
<td>ERMST, SERV1</td>
</tr>
<tr>
<td>DERM1</td>
<td>DERM1</td>
<td>ERMST, DERV1</td>
</tr>
<tr>
<td>IERM1</td>
<td>IERM1</td>
<td>ERMST, IERV1</td>
</tr>
<tr>
<td>SERMN</td>
<td>SERMN</td>
<td>ERMST, SERVN</td>
</tr>
<tr>
<td>DERMN</td>
<td>DERMN</td>
<td>ERMST, DERVN</td>
</tr>
<tr>
<td>SERVN</td>
<td>SERVN</td>
<td>ERMST, DERVN</td>
</tr>
<tr>
<td>DERVN</td>
<td>DERVN</td>
<td>ERMST, DERVN</td>
</tr>
<tr>
<td>ERFIN</td>
<td>ERFIN</td>
<td>None</td>
</tr>
</tbody>
</table>

DRERMSG

C     program DRERMSG
C>> 1988-11-16
C
integer IDELTA
real SX
double precision DX
C
data SX,DX / 1.0E0,2.0D0 /
C
call ERMSG('AAAA',1,0,'Description of error.',',','

ODERMSG

Subprogram AAAA reports error number 1
Description of error.
SX = 1.000000

Subprogram BBBB reports error number 2
Description of 2nd error.
SUBPROGRAM CCCC REPORTS ERROR NO.  3
Description of 3rd error.
  SX =     1.000000
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SUBPROGRAM DDDD REPORTS ERROR NO.  4
Description of 4th error.
  DX =   2.0000000000000000000
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SUBPROGRAM EEEE REPORTS ERROR NO.  5
Testing ERMSET
  IDELTA =    -1
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SUBPROGRAM EEEE REPORTS ERROR NO.  5
Testing ERMSET
  IDELTA =     0
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SUBPROGRAM EEEE REPORTS ERROR NO.  5
Testing ERMSET
  IDELTA =     1
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

SUBPROGRAM EEEE REPORTS ERROR NO.  5
Testing ERMSET
  IDELTA =     2
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
19.3 **Extended Error Message Processor**

**A. Purpose**

The subroutines described here print error messages and diagnostic messages. These routines are intended primarily for use by other library routines. Only that part of the package which allows the user to change the actions taken when library programs use this package are described here.

**B. Usage**

**Program Prototype, Setting or Getting a Message Processor Parameter**

```plaintext
INTEGER MACT(≥3), IDAT(≥1)
CHARACTER*(≥1) TEXT(≥1)
```

Assign values to `MACT()`. In the application here, `IDAT()`, and `TEXT` (with one exception) are not referenced.

```plaintext
CALL MESS(MACT, TEXT, IDAT)
```

**Argument Definitions**

`MACT() [inout]` Used to set or check parameters used by the message processor. Starting at `MACT(1)`, is a list of integer pairs followed by a single `MERET=51`, which terminates the list. If the first integer is > 0, it is an index specifying the parameter one wants to set and the following integer gives the value to be assigned to the parameter. If the first integer is negative, its absolute value specifies a parameter as above, and the value of that parameter is returned in the second integer. A single call can get and/or set as many of these parameters as desired. Below is a list of parameter names (which we recommend for clarity of the code), and their values, which are used as the first integer in an entry. The second integer of the entry is denoted by `Kj`, where `j` is the value of the first integer.

- **MESUNI=10** (K10) Set the scratch file unit number to K10. The default unit for a scratch file is 30. If a scratch file is needed, (only needed here if a table exceeds the line length), and unit 30 cannot be opened as a new scratch file, then units 29, 28, ..., will be tried until an acceptable unit is found. If K10 is set to 0, a scratch unit is assumed not to be available, and tables with long lines will be printed with each line on multiple lines.

- **MEHEAD=11** (K11) Defines the print that surrounds an error message. K11=0 gives nothing, and 1 gives the first 4 characters in `TEXT` repeated 18 times. If this is not used, one gets 72 $'s. (To get a blank line use 1 with TEXT(1) = '    '.)

- **MEDDIG=12** (K12) Set default digits to print for floating point. If K12 > 0, K12 significant digits will be printed, if K12 < 0, -K12 digits will be printed after the decimal point, and if K12 = 0, the default will be used, which is the full machine precision. Setting or getting this value will only work properly if the action is taken by calling `SMESS` or `DMESS` as appropriate; see below. There are separate internal values of this value in `SMESS` and `DMESS`.

- **MEMLIN=13** (K13) Set line length for diagnostic messages to K13. (Default is 128.)

- **MEELIN=14** (K14) Set line length for error messages to K14. (Default is 79.)

- **MEMUNI=15** (K15) Set unit number for diagnostic messages to K15. If K15 = 0 (default), 'print' is used.
MEEUNI=16 (K16) Set unit number for error messages to K16. If K16 = 0 (default), 'print' is used.

MESCRN=17 (K17) Set number of lines to print to standard output before pausing for "go" from user. Default is 0, which never stops.

MEDIAG=18 (K18) Not currently in use by MATH77 routines.

MEMAXE=19 (K19) Set the maximum error value. It is the maximum value seen for 10000*s + 1000*p + i; where s, p, and i are the stop and print levels, and the index on the last error message processed, respectively.

MESTOP=20 (K20) Set the stop level for error messages. If an error message has a stop index > min(K20, 8), the program is stopped after processing the message. The default value is K20=3.

MEPRNT=21 (K21) Set the print level for error messages. If an error message has a print index > K21, or the message is going to stop when finished, information in an error message is processed, else all the actions including printing are skipped. (MESTOP controls stopping.) The default value is K21=3.

MERET=51 End of the list. (Only requires a single integer.)

TEXT [in] Only referenced if MEHEAD is one of the actions. See the description above.

IDAT() [in] Not referenced by the application discussed here.

Getting and Setting the Default for Digits to Print for Floating Point Numbers
As mentioned above, the setting or retrieving of the number of decimal digits, MEDDIG above, requires a call to SMESS for single precision, and a call to DMESS for double precision. Either of these calls can be used to set any of the other parameters also. These calls require the additional declaration of a floating point array FDAT, which must be single precision if SMESS is called and double precision if DMESS is called.

CALL SMESS(MACT, TEXT, IDAT, FDAT)

CALL DMESS(MACT, TEXT, IDAT, FDAT)

As above, the only argument actually used is MACT() (and perhaps TEXT). The remaining arguments are present for other applications of this package.

C. Examples and Remarks
The program PMESS described in [1] was used to transform the data in DRSMESS.ERR into statements included in the listing for DRSMESS.FOR. The output is given in ODSMESS. This illustrates how error messages can be generated, and how the length of the output line can be changed from the default (which is 79) to 40.

D. Functional Description
The routines mentioned here are called by a number of library routines for the printing of both error messages and diagnostic information. The part of the package described here allows one to
alter this output. Complete documentation on MESS, SMESS, and DMESS can be found in ref. [1].

Reference

E. Error Procedures and Restrictions
Use of indices other than those described here may cause actions as described in ref. [1], or result in a "STOP" being executed in MESS.

F. Supporting Information
The source language is ANSI Fortran 77.

<table>
<thead>
<tr>
<th>Program Unit and Entry Name</th>
<th>External References</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESS</td>
<td>MESS, R1MACH</td>
</tr>
<tr>
<td>SMESS</td>
<td>MESS, D1MACH</td>
</tr>
</tbody>
</table>

DRSMESS.ERR

TSMESS$B
Description of error. (With one real)  FDATA1 = $F.$E
Description of 2nd error. (With no data) $E
Description of 3rd error. (With one integer)  IDATA1 = $I.$E
Description of 4th error. (With two integers and two reals) $N
IDATA1 = $I, IDATA2 = $I, FDATA2 = $F, FDATA3 = $F.$E
DRSMESS.FOR

program DRSMESS

integer MACT(5), IDAT(3), ILOC(4), MACT1(3)
real FDAT(3)
logical LIN40

c parameter (MERET =51)
parameter (MEEMES =52)
parameter (MESTOP =20)
parameter (MEELIN =14)
integer MLOC(4)

c ------ Start of code generated by PMESS from DRSMESS.ERR ----------
character MTEXT(281)
parameter (LTXTAA = 1)
character MTXTAA*(8)
equivalence (MTEXT(LTXTAA), MTXTAA)
parameter (LTXTAB = 9)
character MTXTAB*(54)
equivalence (MTEXT(LTXTAB), MTXTAB)
parameter (LTXTAC = 63)
character MTXTAC*(43)
equivalence (MTEXT(LTXTAC), MTXTAC)
parameter (LTXTAD = 106)
character MTXTAD*(61)
equivalence (MTEXT(LTXTAD), MTXTAD)
parameter (LTXTAE = 167)
character MTXTAE*(62)
equivalence (MTEXT(LTXTAE), MTXTAE)
parameter (LTXTAF = 229)
character MTXTAF*(53)
equivalence (MTEXT(LTXTAF), MTXTAF)
DATA MTEXTAA/'TSMESS$B'/,
1MTXTAB/'Description of error. (With one real) FDATA1 = $F.$E'/,
2MTXTAC/'Description of 2nd error. (With no data)$E'/,
3MTXTAD/ 4'Description of 3rd error. (With one integer) IDATA1 = $I.$E'/,
5MTXTAE/ 6'Description of 4th error. (With two integers and two reals)$N'/,
7MTXTAF/'IDATA1 = $I, IDATA2 = $I, FDATA2 = $F, FDATA3 = $F.$E'/

--------- End of code generated by PMESS from DRSMESS.ERR ---------
data MLOC /LTXTAB,LTXTAC,LTXTAD,LTXTAE/

c

data MACT / MEEMES,25,0,0, MERET /
data MACT1 / MEELIN, 40, MERET /
data FDAT / 1.7E-12, -12.3456789, 3.1415926 /
data IDAT / 17, -178, 4 /
data ILOC / 1, 1, 2, 2 /

c LIN40 = .false.
c

Loop to print error messages.

19.3-4 Extended Error Message Processor
10 do 100 I = 1, 4
   MACT(3) = I
   MACT(4) = MLOC(I)
   call SMESS(MACT, MTEXT, IDAT(ILOC(I)), FDAT(ILOC(I)))
100 continue
   if (LIN40) stop
   call MESS(MACT1, MTEXT, IDAT)
   LIN40 = .true.
   go to 10
end

ODSMESS

$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 1
Description of error. (With one real) FDATA1 = 1.7000000E-12.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 2
Description of 2nd error. (With no data)
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 3
Description of 3rd error. (With one integer) IDATA1 = -178.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 4
Description of 4th error. (With two integers and two reals)
IDATA1 = -178, IDATA2 = 4, FDATA2 = -12.345680, FDATA3 = 3.1415930.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 1
Description of error. (With one real) FDATA1 = 1.7000000E-12.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 2
Description of 2nd error. (With no data)
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 3
Description of 3rd error. (With one integer) IDATA1 = -178.
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
TSMESS reports error: Stop level = 2, Print level = 5, Error index = 4
Description of 4th error. (With two integers and two reals)
IDATA1 = -178, IDATA2 = 4, FDATA2 = -12.345680, FDATA3 = 3.1415930.
integer) IDATA1 = -178.

TSMESS reports error: Stop level = 2,
Print level = 5, Error index = 4
Description of 4th error. (With two
integers and two reals)
IDATA1 = -178, IDATA2 = 4, FDATA2 =
-12.345680, FDATA3 = 3.1415930.