FORTRAN IV LIBRARY
TECHNICAL MANUAL
for
SDS SIGMA 5/7 COMPUTERS

90 15 24A

March 1969

SDS
SCIENTIFIC DATA SYSTEMS/701 South Aviation Boulevard/El Segundo, California 90245
## RELATED PUBLICATIONS

<table>
<thead>
<tr>
<th>Title</th>
<th>Publication No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDS Sigma 5/7 Mathematical Routines Technical Manual</td>
<td>90 09 06</td>
</tr>
<tr>
<td>SDS Sigma 5/7 FORTRAN IV Reference Manual</td>
<td>90 09 56</td>
</tr>
<tr>
<td>SDS Sigma 5/7 FORTRAN IV Operations Manual</td>
<td>90 11 43</td>
</tr>
<tr>
<td>SDS Sigma 5/7 Batch Processing Monitor Reference Manual</td>
<td>90 09 54</td>
</tr>
<tr>
<td>SDS Sigma 5/7 Batch Processing Monitor Operations Manual</td>
<td>90 11 98</td>
</tr>
<tr>
<td>SDS Sigma 5/7 Symbol and Meta-Symbol Reference Manual</td>
<td>90 09 52</td>
</tr>
<tr>
<td>SDS Sigma 5 Computer Reference Manual</td>
<td>90 09 59</td>
</tr>
<tr>
<td>SDS Sigma 7 Computer Reference Manual</td>
<td>90 09 50</td>
</tr>
</tbody>
</table>

### NOTICE

The specifications of the software system described in this publication are subject to change without notice. The availability or performance of some features may depend on a specific configuration of equipment such as additional tape units or larger memory. Customers should consult their SDS sales representative for details.
## CONTENTS

1. INTRODUCTION
   - Library Classification .......................... 1
   - Operating Environment .......................... 2
   - Register Conventions .......................... 2
   - Library Ordering .......................... 2
   - Reentrancy .................................. 3

2. PROGRAM DESCRIPTIONS .......................... 10

## TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommended Order for SDS Sigma 5/7</td>
<td>3</td>
</tr>
<tr>
<td>FORTRAN IV Library Routines</td>
<td>3</td>
</tr>
<tr>
<td>Library Routines</td>
<td>4</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

LIBRARY CLASSIFICATION

The SDS Sigma 5/7 FORTRAN IV Library can be classified into five categories:

1. Standard drivers
2. Standard evaluators
3. Standard subroutines
4. Nonstandard evaluators
5. System routines (nonstandard)

In order to understand this classification, it is necessary to define the following terms.

INTRINSIC FUNCTION A library function that is generated in-line (or partially in-line) by the compiler. An example of "partially in-line" is SQRT, which is recognized by the compiler as an intrinsic function. This does not mean that it generates square roots in-line, but that it calls a special routine (not named SQRT) to do the work. Thus, the distinguishing characteristic of an intrinsic function is that it does not generate a call on a routine with the same name that the user wrote in his statement.

BASIC EXTERNAL FUNCTION A library function whose type (integer, real, etc.) is known by the compiler, but which is called externally with the same name the user wrote. In the case of a function like MOD (which is an integer function and also begins with M), the compiler does not have to know anything about the function.

FORTRAN NAME A name that begins with a letter and contains only letters and digits, i.e., a name that can be used in a FORTRAN source program. Basic external functions all have FORTRAN names.

NON-FORTRAN NAME A name that is not a FORTRAN name. Any routine that is called by the compiler without having been specifically named by the user must have a non-FORTRAN name. Otherwise, it might conflict with some FORTRAN name used elsewhere in the program. Sigma 5/7 FORTRAN IV has a convention for non-FORTRAN names that aids understanding of the routines in the library. Each such name begins with a digit, as follows:

9 - Primary routine that can be referenced directly by compiled code or might reasonably be referenced by a user. For example, 9SQRT, 9SETUPN.
8 - A storage cell or area referenced by more than one library routine. For example, 8EOFEXIT, 8BCDBUF.
7 - Secondary routine, typically referenced only by other library routines. For example, 7EOFABRT, 7EXP.

6 - A storage cell that is external only for the purposes of reentrance, and could otherwise be local to one routine. For example, 6EOFCALL.

Other non-FORTRAN names, such as those beginning with F: or M: are Monitor names and are not part of the FORTRAN library.

STANDARD CALLING SEQUENCE A means of calling subprograms and passing arguments that is standard to all FORTRAN-compiled routines and, therefore, to all routines with FORTRAN names. A subprogram that is called with a standard calling sequence must begin with a standard receiving sequence, which involves the use of one of the 9SETUP routines (see the FORTRAN IV Operations Manual, SDS 90 11 43).

NONSTANDARD CALLING SEQUENCE Any means of calling a subprogram and passing arguments that does not obey the rules of a standard name routine with a standard calling sequence; therefore, nonstandard calling sequences are used only to call routines with non-FORTRAN names. It is permissible to call a non-FORTRAN name with a standard calling sequence, but not conversely. Thus, it is possible to design a FORTRAN system in which there are no nonstandard calling sequences, but it would be inefficient to do so. Therefore, the rule in SDS Sigma 5/7 FORTRAN IV is FORTRAN name <-> standard calling sequence; non-FORTRAN name <-> nonstandard calling sequence.

NONSTANDARD EVALUATOR A closed library routine with a non-FORTRAN name and called with a nonstandard calling sequence that evaluates some mathematical function. This includes the Mathematical Library described in the Sigma 5/7 Mathematical Routines Technical Manual, SDS 90 09 06. In Sigma 5/7 FORTRAN IV, all of the intrinsic functions either call a nonstandard evaluator or are generated in-line.

STANDARD DRIVER A closed library routine with a FORTRAN name, and therefore called with a standard calling sequence, that does not perform any significant computation but merely calls on a nonstandard evaluator to do so. Thus it acts as an interface between a standard calling sequence and a nonstandard evaluator. In Sigma 5/7 FORTRAN IV, a standard driver is used only if its name has been declared EXTERNAL (which changes it from intrinsic to basic external).

STANDARD EVALUATOR A closed library routine with a FORTRAN name, and therefore called with a standard calling sequence, that does evaluate some mathematical (although usually trivial) function, rather than calling a nonstandard evaluator. Standard evaluators, like the standard drivers, are called only if declared EXTERNAL. Thus all basic external functions are either standard drivers or standard evaluators. The standard drivers correspond to
those functions that, when intrinsic, call a nonstandard evaluator. The standard evaluators correspond to those functions that, when intrinsic, are generated in-line.

SYSTEM ROUTINE A general term to describe routines that are provided by the system without the specific knowledge of the user. For example, aREWIND statement generates a call on the system routine 9REWIND. System routines always have non-FORTRAN names.

No routine should have both a non-FORTRAN name and a FORTRAN name attached to it, because a call on the non-FORTRAN name (unknown to the user) would also bring in the FORTRAN name, which might conflict with a name in his program. Similarly, no library routine should have two FORTRAN names, nor should any library routine call a routine with a FORTRAN name.

STANDARD SUBROUTINE A subroutine (as opposed to function) with a FORTRAN name and a standard calling sequence. Standard subroutine is to subroutine as standard evaluator is to function. Examples are BUFFER IN, EOFSET, EXIT. A routine like SSWATCH is a hybrid; it can be used both as a standard subroutine and as a standard evaluator (function).

In summary, among the five categories of library routines there are many standard drivers and evaluators, but they are infrequently used (only when declared EXTERNAL). There are only a few standard subroutines. Most of the library routines called by a typical program are nonstandard evaluators and system routines.

Table 1 lists the FORTRAN IV library routines, by catalog number. The remainder of this manual contains descriptions of these routines. The mathematical routines are described only in general; detailed descriptions may be found in the Sigma 5/7 Mathematical Routines Technical Manual.

**OPERATING ENVIRONMENT**

The Sigma 5/7 FORTRAN IV Library is designed to run under the Batch Processing Monitor. In doing so, it makes use of a number of Monitor features, including:

1. A variety of CALs, mainly to do input or output. Each library routine contains the names of all CALs it uses near the end of its listing.
2. Automatic loader-generation of DCBs that are REFed.
4. The position of particular fields within DCBs.
5. The position of particular fields within the TCB (Task Control Block).
6. The TCB pointer in register 0. (This must never be destroyed.)
7. The nineteen words of information provided (via a pointer in register 1) when a trap occurs over which the library has requested control.

In addition to these Monitor features, the library also requires initialization before each job. This is performed by 9INITIAL, which must be called at the beginning of every main program. It initializes the following:

1. Miscellaneous library triggers, such as the sense lights and the end-of-file exit.
2. Trap control, set up to process floating overflow, ignore fixed point overflow, and abort to the Monitor on other illegal situations (such as unimplemented instructions or memory protection violation).
3. The floating control indicators. These are set as follows:
   - FS = 0 Do not trap on loss of significance.
   - FZ = 0 Do not trap on floating underflow.
   - FN = 0 Post-normalize all results. Unnormalized floating-point numbers are not permitted in the system.

These floating control status settings are depended on throughout the system. If they have to be changed temporarily (such as in 9DTOI) they must be immediately restored. Otherwise mathematical routines may produce meaningless results and traps may occur that are not provided for.

**REGISTER CONVENTIONS**

Register 0 contains the TCB pointer, which must never be destroyed by any routine, whether library or user.

Standard subprograms (called with a standard calling sequence) are allowed to destroy any of the other registers.

Nonstandard (system) routines vary widely in their register usage. In general, however, index registers 1, 3, 5, and 7 are preserved by system routines and by the math library.

**LIBRARY ORDERING**

It is often desirable to arrange a library in an order where no routine references any routine that has already appeared. In this way, a selective library load can be performed in one pass. Table 1 shows the recommended ordering of the FORTRAN library to accomplish this one-pass load objective. The ordering maintains catalog-number order wherever possible. Only the last three digits of the catalog number are shown; the first three digits are always 705.

Table 2 lists all library routines in catalog number sequence. The descriptions of these routines, which are given in Section 2, are also in sequence by catalog number. Descriptions for 9IFD(705054), 9INPUT(705071), and 9RDDISC(705079) have been omitted. They will be included in the next edition of this manual, which will describe the Sigma 5/7 Extended FORTRAN IV release.
Table 1. Recommended Order for SDS Sigma 5/7 FORTRAN IV Library Routines

<table>
<thead>
<tr>
<th>(By last 3 digits of catalog number)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>101 - 243</td>
<td></td>
</tr>
<tr>
<td>024 - 035</td>
<td></td>
</tr>
<tr>
<td>038 - 045</td>
<td></td>
</tr>
<tr>
<td>048</td>
<td></td>
</tr>
<tr>
<td>052</td>
<td></td>
</tr>
<tr>
<td>036 - 037</td>
<td></td>
</tr>
<tr>
<td>046 - 047</td>
<td></td>
</tr>
<tr>
<td>011 - 023</td>
<td></td>
</tr>
<tr>
<td>001 - 010</td>
<td></td>
</tr>
<tr>
<td>053 - 056</td>
<td></td>
</tr>
<tr>
<td>058</td>
<td></td>
</tr>
<tr>
<td>057</td>
<td></td>
</tr>
<tr>
<td>059 - 061</td>
<td></td>
</tr>
<tr>
<td>063</td>
<td></td>
</tr>
<tr>
<td>062</td>
<td></td>
</tr>
<tr>
<td>064 - 065</td>
<td></td>
</tr>
<tr>
<td>067 - 090</td>
<td></td>
</tr>
<tr>
<td>066</td>
<td></td>
</tr>
<tr>
<td>091</td>
<td></td>
</tr>
<tr>
<td>051</td>
<td></td>
</tr>
<tr>
<td>092 - 094</td>
<td></td>
</tr>
</tbody>
</table>

**REEENTRY**

All FORTRAN IV library routines are time-sharing (task) reentrant. They are not guaranteed to be real-time (interrupt) reentrant, although some of them are. However, the Real-Time FORTRAN Library does contain completely interruptable reentrant routines.

The distinction between these forms of reentrance is that in a time-sharing environment, there are a fixed number of users (tasks) active at any one time, and the Monitor knows how many there are.

It can produce duplicate copies of all context storage (CSECT 0) for each user. The library coding must then follow two rules: it must not modify instructions (i.e., store into a CSECT 1 region) and the context area that it does store into must be referenced externally. This context area in the FORTRAN IV library consists of the assemblies whose names begin with 8 or 6 (e.g., 8TEDIT). If these areas are replicated for each user, the library is completely reentrant.

Interrupt reentrance, however, is more complicated. There is no limit on the number of interrupts that may occur, so a routine may be reentered any number of times. Each time it is reentered, it needs a complete set of context storage. These storage areas cannot be allocated in advance (as they can in a time-sharing system); each needs to be allocated whenever a routine is reentered because of an interrupt. Thus the Real-Time FORTRAN Library will include a means of obtaining push-down stack storage to be used by routines that are reentered.
<table>
<thead>
<tr>
<th>Cat. No.</th>
<th>Name of Routine</th>
<th>Other Entries</th>
<th>Size</th>
<th>Description of Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>705001</td>
<td>9ALOG</td>
<td>7ALOG</td>
<td>79</td>
<td>Real natural logarithm</td>
</tr>
<tr>
<td>705002</td>
<td>9DLOG</td>
<td>7DLOG</td>
<td>116</td>
<td>Double precision natural logarithm</td>
</tr>
<tr>
<td>705003</td>
<td>9EXP</td>
<td>7EXP1</td>
<td>59</td>
<td>Real exponential</td>
</tr>
<tr>
<td>705004</td>
<td>9DEXP</td>
<td>7DEXP1</td>
<td>87</td>
<td>Double precision exponential</td>
</tr>
<tr>
<td>705005</td>
<td>9SIN 9COS</td>
<td></td>
<td>90</td>
<td>Real sine and cosine</td>
</tr>
<tr>
<td>705006</td>
<td>9DSIN 9DCOS</td>
<td></td>
<td>133</td>
<td>Double precision sine and cosine</td>
</tr>
<tr>
<td>705007</td>
<td>9ATAN1 9ATAN2</td>
<td></td>
<td>91</td>
<td>Real arctangent</td>
</tr>
<tr>
<td>705008</td>
<td>9DATAN1 9DATAN2</td>
<td></td>
<td>121</td>
<td>Double precision arctangent</td>
</tr>
<tr>
<td>705009</td>
<td>9SORT</td>
<td></td>
<td>38</td>
<td>Real square root</td>
</tr>
<tr>
<td>705010</td>
<td>9DSQRT</td>
<td>9COSH</td>
<td>63</td>
<td>Double precision square root</td>
</tr>
<tr>
<td>705011</td>
<td>9SINH</td>
<td></td>
<td>78</td>
<td>Real hyperbolic sine and cosine</td>
</tr>
<tr>
<td>705012</td>
<td>9DSINH 9DCOSH</td>
<td></td>
<td>81</td>
<td>Double precision hyperbolic sine and cosine</td>
</tr>
<tr>
<td>705013</td>
<td>9TANH</td>
<td></td>
<td>60</td>
<td>Real hyperbolic tangent</td>
</tr>
<tr>
<td>705014</td>
<td>9DANTANH</td>
<td></td>
<td>68</td>
<td>Double precision hyperbolic tangent</td>
</tr>
<tr>
<td>705015</td>
<td>9ASIN 9ACOS</td>
<td></td>
<td>26</td>
<td>Real arcsine and arccosine</td>
</tr>
<tr>
<td>705016</td>
<td>9DASIN 9DACOS</td>
<td></td>
<td>65</td>
<td>Double precision arcsine and arccosine</td>
</tr>
<tr>
<td>705017</td>
<td>9TAN</td>
<td></td>
<td>70</td>
<td>Real tangent</td>
</tr>
<tr>
<td>705018</td>
<td>9DTAN</td>
<td></td>
<td>110</td>
<td>Double precision tangent</td>
</tr>
<tr>
<td>705019</td>
<td>9ALOG10</td>
<td></td>
<td>19</td>
<td>Real common logarithm</td>
</tr>
<tr>
<td>705020</td>
<td>9DLOG10</td>
<td></td>
<td>21</td>
<td>Double precision common logarithm</td>
</tr>
<tr>
<td>705021</td>
<td>9PWRI</td>
<td></td>
<td>37</td>
<td>Integer raised to an integer power</td>
</tr>
<tr>
<td>705022</td>
<td>9PWRII</td>
<td></td>
<td>35</td>
<td>Real raised to an integer power</td>
</tr>
<tr>
<td>705023</td>
<td>9PWREDI</td>
<td></td>
<td>35</td>
<td>Double precision raised to an integer power</td>
</tr>
<tr>
<td>705024</td>
<td>9PWRCI</td>
<td></td>
<td>67</td>
<td>Complex raised to an integer power</td>
</tr>
<tr>
<td>705025</td>
<td>9PWRRK</td>
<td></td>
<td>74</td>
<td>Double complex raised to an integer power</td>
</tr>
<tr>
<td>705026</td>
<td>9PWRRD</td>
<td></td>
<td>57</td>
<td>Real raised to a real power</td>
</tr>
<tr>
<td>705027</td>
<td>9PWRRCC</td>
<td></td>
<td>57</td>
<td>Double precision raised to a double precision power</td>
</tr>
<tr>
<td>705028</td>
<td>9PWRRKK</td>
<td></td>
<td>40</td>
<td>Complex raised to a complex power</td>
</tr>
<tr>
<td>705029</td>
<td>9PWRRKK</td>
<td></td>
<td>41</td>
<td>Double complex raised to a double complex power</td>
</tr>
<tr>
<td>705030</td>
<td>9CADD</td>
<td>9CSUB</td>
<td>6</td>
<td>Complex add and subtract</td>
</tr>
<tr>
<td>705031</td>
<td>9CMUL</td>
<td></td>
<td>10</td>
<td>Complex multiply</td>
</tr>
<tr>
<td>705032</td>
<td>9CDIV</td>
<td></td>
<td>19</td>
<td>Complex divide</td>
</tr>
<tr>
<td>705033</td>
<td>9KADD 9KSUB</td>
<td></td>
<td>6</td>
<td>Double complex add and subtract</td>
</tr>
<tr>
<td>705034</td>
<td>9KMU</td>
<td></td>
<td>10</td>
<td>Double complex multiply</td>
</tr>
<tr>
<td>705035</td>
<td>9KDIV</td>
<td></td>
<td>17</td>
<td>Double complex divide</td>
</tr>
<tr>
<td>705036</td>
<td>9CLOG 7CLOG</td>
<td></td>
<td>56</td>
<td>Complex natural logarithm</td>
</tr>
<tr>
<td>705037</td>
<td>9CDLOG 7CDLOG</td>
<td></td>
<td>53</td>
<td>Double complex natural logarithm</td>
</tr>
<tr>
<td>705038</td>
<td>9CEXP 7CEXP</td>
<td></td>
<td>20</td>
<td>Complex exponential</td>
</tr>
<tr>
<td>705039</td>
<td>9CDEXP 7CDEXP</td>
<td></td>
<td>28</td>
<td>Double complex exponential</td>
</tr>
<tr>
<td>705040</td>
<td>9CSIN 9CCOS</td>
<td>9CSINH 9CCCOSH</td>
<td>86</td>
<td>Complex sine, cosine, hyperbolic sine, and hyperbolic cosine</td>
</tr>
<tr>
<td>705041</td>
<td>9CDSIN 9CDCOS</td>
<td>9CDSINH 9CDCOSH</td>
<td>112</td>
<td>Double complex sine, cosine, hyperbolic sine, and hyperbolic cosine</td>
</tr>
<tr>
<td>705042</td>
<td>9CTAN</td>
<td>9CTANH</td>
<td>52</td>
<td>Complex tangent and hyperbolic tangent</td>
</tr>
<tr>
<td>705043</td>
<td>9CDTAN</td>
<td>9CDTANH</td>
<td>59</td>
<td>Double complex tangent and hyperbolic tangent</td>
</tr>
<tr>
<td>705044</td>
<td>9CATAN</td>
<td></td>
<td>76</td>
<td>Complex arctangent</td>
</tr>
<tr>
<td>705045</td>
<td>9CDATAN</td>
<td></td>
<td>104</td>
<td>Double complex arctangent</td>
</tr>
</tbody>
</table>
Table 2. Library Routines (cont.)

<table>
<thead>
<tr>
<th>Cat. No.</th>
<th>Name of Routine</th>
<th>Other Entries</th>
<th>Size</th>
<th>Description of Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>705046</td>
<td>9CSQRT</td>
<td>9CABS</td>
<td>88</td>
<td>Complex square root and absolute value (real modulus)</td>
</tr>
<tr>
<td>705047</td>
<td>9CDSQRT</td>
<td>9CDABS</td>
<td>92</td>
<td>Double complex square root and absolute value (double precision modulus)</td>
</tr>
<tr>
<td>705048</td>
<td>9CASIN</td>
<td>9CACOS, 9CDASIN</td>
<td>91</td>
<td>Complex arcsine and arccosine; double complex arcsine and arccosine</td>
</tr>
<tr>
<td>705051</td>
<td>8T0</td>
<td></td>
<td>17</td>
<td>FORTRAN library temp area</td>
</tr>
<tr>
<td>705052</td>
<td>7SIN</td>
<td></td>
<td>85</td>
<td>Special SIN/COS/EXP calculations used by various complex functions</td>
</tr>
<tr>
<td>705053</td>
<td>9IFR</td>
<td></td>
<td>21</td>
<td>Real approximate equality test</td>
</tr>
<tr>
<td>705054</td>
<td>9IFD</td>
<td></td>
<td></td>
<td>Double precision approximate equality test</td>
</tr>
<tr>
<td>705055</td>
<td>9ITOD</td>
<td>9ITOR</td>
<td>7</td>
<td>Integer to double precision and integer to real conversions</td>
</tr>
<tr>
<td>705056</td>
<td>9DTOI</td>
<td>9RTOI</td>
<td>8</td>
<td>Double precision to integer and real to integer conversions</td>
</tr>
<tr>
<td>705057</td>
<td>9DTOR</td>
<td></td>
<td>24</td>
<td>Double precision to real conversion</td>
</tr>
<tr>
<td>705058</td>
<td>9KTOC</td>
<td></td>
<td>8</td>
<td>Double complex to complex conversion</td>
</tr>
<tr>
<td>705059</td>
<td>9SETUP0</td>
<td></td>
<td>2</td>
<td>Set up zero arguments</td>
</tr>
<tr>
<td>705060</td>
<td>9SETUP1</td>
<td></td>
<td>7</td>
<td>Set up one argument</td>
</tr>
<tr>
<td>705061</td>
<td>9SETUP2</td>
<td></td>
<td>13</td>
<td>Set up two arguments</td>
</tr>
<tr>
<td>705062</td>
<td>9SETUPN</td>
<td>9SETUPM, 7SET</td>
<td>14</td>
<td>Set up a fixed number of arguments or a variable number with maximum</td>
</tr>
<tr>
<td>705063</td>
<td>9SETUPV</td>
<td></td>
<td>11</td>
<td>Set up a variable number of arguments</td>
</tr>
<tr>
<td>705064</td>
<td>9INITIAL</td>
<td></td>
<td>47</td>
<td>Run-time initialization</td>
</tr>
<tr>
<td>705065</td>
<td>9ERROR</td>
<td></td>
<td>110</td>
<td>Math library error reporting</td>
</tr>
<tr>
<td>705066</td>
<td>7ERROR</td>
<td></td>
<td>144</td>
<td>Run-time error reporting</td>
</tr>
<tr>
<td>705067</td>
<td>9BCDREAD</td>
<td>9READ, 7BCDREAD</td>
<td>48</td>
<td>BCD read</td>
</tr>
<tr>
<td>705068</td>
<td>9BCDWRIT</td>
<td>9PRINT, 7BCDWRIT</td>
<td>57</td>
<td>BCD write</td>
</tr>
<tr>
<td>705069</td>
<td>9BINREAD</td>
<td>9BINWRIT, 7BINREAD</td>
<td>225</td>
<td>Binary read and write</td>
</tr>
<tr>
<td>705070</td>
<td>9DECODE</td>
<td>9ENCODE</td>
<td>35</td>
<td>Memory-to-memory data conversion</td>
</tr>
<tr>
<td>705071</td>
<td>9INPUT</td>
<td></td>
<td></td>
<td>Self-identified (NAME LIST) input</td>
</tr>
<tr>
<td>705072</td>
<td>9EDIT</td>
<td>9OEDIT</td>
<td>1407</td>
<td>FORMAT scan</td>
</tr>
<tr>
<td>705073</td>
<td>9IOLUSA</td>
<td></td>
<td>35</td>
<td>Transmit unsubscripted arrays in an I/O list</td>
</tr>
<tr>
<td>705074</td>
<td>9IODATA</td>
<td>9DATA, 9ENDIOL</td>
<td>132</td>
<td>Transmit individual items in an I/O list</td>
</tr>
<tr>
<td>705075</td>
<td>9REWIND</td>
<td></td>
<td>14</td>
<td>Rewind sequential files</td>
</tr>
<tr>
<td>705076</td>
<td>9KSPACE</td>
<td></td>
<td>38</td>
<td>Backspace one logical record</td>
</tr>
<tr>
<td>705077</td>
<td>9ENDFILE</td>
<td></td>
<td>14</td>
<td>Write end-of-file</td>
</tr>
<tr>
<td>705078</td>
<td>7EOFABRT</td>
<td></td>
<td>33</td>
<td>End-of-file abort</td>
</tr>
<tr>
<td>705079</td>
<td>9DDISC</td>
<td>9WRDISC</td>
<td>64</td>
<td>Read and write disc (random access)</td>
</tr>
<tr>
<td>705080</td>
<td>7UNITADR</td>
<td></td>
<td>8</td>
<td>Find I/O unit DCB address</td>
</tr>
<tr>
<td>705081</td>
<td>9ASFORM</td>
<td></td>
<td>29</td>
<td>Assigned FORMAT</td>
</tr>
<tr>
<td>705082</td>
<td>9ASGOTO</td>
<td></td>
<td>24</td>
<td>Assigned GO TO</td>
</tr>
<tr>
<td>705083</td>
<td>9FSWITCH</td>
<td></td>
<td>24</td>
<td>Test sense switch</td>
</tr>
<tr>
<td>705084</td>
<td>9SNSLITE</td>
<td>9IFSLITE</td>
<td>30</td>
<td>Set and test sense lights</td>
</tr>
<tr>
<td>Cat. No.</td>
<td>Name of Routine</td>
<td>Other Entries</td>
<td>Size</td>
<td>Description of Routine</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>---------------</td>
<td>------</td>
<td>------------------------</td>
</tr>
<tr>
<td>705085</td>
<td>9IFOVFL</td>
<td></td>
<td>4</td>
<td>Test for floating overflow</td>
</tr>
<tr>
<td>705086</td>
<td>9UNDEFLB</td>
<td></td>
<td>13</td>
<td>Undefined label abort</td>
</tr>
<tr>
<td>705087</td>
<td>9PAUSE</td>
<td></td>
<td>20</td>
<td>PAUSE</td>
</tr>
<tr>
<td>705088</td>
<td>9STOP</td>
<td>7STOP</td>
<td>26</td>
<td>STOP (terminate execution)</td>
</tr>
<tr>
<td>705089</td>
<td>7BINDEC</td>
<td></td>
<td>13</td>
<td>Binary to decimal for messages</td>
</tr>
<tr>
<td>705090</td>
<td>7GETMODE</td>
<td></td>
<td>12</td>
<td>Argument mode calculation</td>
</tr>
<tr>
<td>705091</td>
<td>8T1</td>
<td>8T2, 8T3, 8T4, 8T5, 8T6, 8T7, 8T8, 8T9, 8T10, 8T11, 8T12, 8T13, 8T14, 8T15, 8T16, 8T17</td>
<td>1</td>
<td>Additional names for library temps</td>
</tr>
<tr>
<td>705092</td>
<td>8TINIT</td>
<td>8ABORTEX, 8ABRTSEV, 8EOFEXIT, 8ERREXIT, 8FLOWTRG, 8IOTRIG, 8SENLITE</td>
<td>14</td>
<td>Temps for 9INITIAL</td>
</tr>
<tr>
<td>705093</td>
<td>8TERROR</td>
<td>8IODLINK, 8MSGBUF, 8TALPHA, 8TBIOTA</td>
<td>52</td>
<td>Temps for 7ERROR</td>
</tr>
<tr>
<td>705094</td>
<td>8TEDIT</td>
<td>6BINBUF, 6DATLINK, 6DCBNAME, 6EEFLAG, 6EOFCALL, 6EOFJADR, 6EOFUADR, 6EOFUTRG, 6NRELMNTS, 6RECSIZE, 8BCDBUF, 8BUFORG, 8DCBADDR, 8ENDIOL, 8NPTERM, 8IODADDR, 8IODTYPE, 8IoENLOC, 8UNITNAME, 8UNITVAL</td>
<td>151</td>
<td>Temps for 1/O</td>
</tr>
<tr>
<td>705101</td>
<td>ALOG</td>
<td></td>
<td>5</td>
<td>Driver for 9ALOG</td>
</tr>
<tr>
<td>705102</td>
<td>DLOG</td>
<td></td>
<td>5</td>
<td>Driver for 9DLOG</td>
</tr>
<tr>
<td>705103</td>
<td>EXP</td>
<td></td>
<td>5</td>
<td>Driver for 9EXP</td>
</tr>
<tr>
<td>705104</td>
<td>DEXP</td>
<td></td>
<td>5</td>
<td>Driver for 9DEXP</td>
</tr>
<tr>
<td>705105</td>
<td>SIN</td>
<td></td>
<td>5</td>
<td>Driver for 9SIN</td>
</tr>
</tbody>
</table>

Table 2. Library Routines (cont.)
<table>
<thead>
<tr>
<th>Cat. No.</th>
<th>Name of Routine</th>
<th>Other Entries</th>
<th>Size</th>
<th>Description of Routine</th>
</tr>
</thead>
<tbody>
<tr>
<td>705106</td>
<td>DSIN</td>
<td>5</td>
<td>Driver for 9DSIN</td>
<td></td>
</tr>
<tr>
<td>705107</td>
<td>ATAN</td>
<td>11</td>
<td>Driver for 9ATAN1 and 9ATAN2</td>
<td></td>
</tr>
<tr>
<td>705108</td>
<td>DATAN</td>
<td>11</td>
<td>Driver for 9DATAN1 and 9DATAN2</td>
<td></td>
</tr>
<tr>
<td>705109</td>
<td>SQRT</td>
<td>5</td>
<td>Driver for 9SQRT</td>
<td></td>
</tr>
<tr>
<td>705110</td>
<td>DSGRT</td>
<td>5</td>
<td>Driver for 9DSQRT</td>
<td></td>
</tr>
<tr>
<td>705111</td>
<td>SINH</td>
<td>5</td>
<td>Driver for 9SINH</td>
<td></td>
</tr>
<tr>
<td>705112</td>
<td>DSINH</td>
<td>5</td>
<td>Driver for 9DSINH</td>
<td></td>
</tr>
<tr>
<td>705113</td>
<td>TANH</td>
<td>5</td>
<td>Driver for 9TANH</td>
<td></td>
</tr>
<tr>
<td>705114</td>
<td>DTANH</td>
<td>5</td>
<td>Driver for 9DTANH</td>
<td></td>
</tr>
<tr>
<td>705115</td>
<td>ASIN</td>
<td>5</td>
<td>Driver for 9ASIN</td>
<td></td>
</tr>
<tr>
<td>705116</td>
<td>DASIN</td>
<td>5</td>
<td>Driver for 9DASIN</td>
<td></td>
</tr>
<tr>
<td>705117</td>
<td>TAN</td>
<td>5</td>
<td>Driver for 9TAN</td>
<td></td>
</tr>
<tr>
<td>705118</td>
<td>DTAI</td>
<td>5</td>
<td>Driver for 9DTAN</td>
<td></td>
</tr>
<tr>
<td>705119</td>
<td>ALOG10</td>
<td>5</td>
<td>Driver for 9ALOG10</td>
<td></td>
</tr>
<tr>
<td>705120</td>
<td>DLOG10</td>
<td>5</td>
<td>Driver for 9DLOG10</td>
<td></td>
</tr>
<tr>
<td>705121</td>
<td>ACOS</td>
<td>5</td>
<td>Driver for 9ACOS</td>
<td></td>
</tr>
<tr>
<td>705122</td>
<td>ATAN2</td>
<td>11</td>
<td>Driver for 9ATAN1 and 9ATAN2</td>
<td></td>
</tr>
<tr>
<td>705123</td>
<td>COS</td>
<td>5</td>
<td>Driver for 9COS</td>
<td></td>
</tr>
<tr>
<td>705124</td>
<td>COSH</td>
<td>5</td>
<td>Driver for 9COSH</td>
<td></td>
</tr>
<tr>
<td>705125</td>
<td>DACOS</td>
<td>5</td>
<td>Driver for 9DACOS</td>
<td></td>
</tr>
<tr>
<td>705126</td>
<td>DATAN2</td>
<td>11</td>
<td>Driver for 9DATAN1 and 9DATAN2</td>
<td></td>
</tr>
<tr>
<td>705127</td>
<td>DCOS</td>
<td>5</td>
<td>Driver for 9DCOS</td>
<td></td>
</tr>
<tr>
<td>705128</td>
<td>DCOSH</td>
<td>5</td>
<td>Driver for 9DCOSH</td>
<td></td>
</tr>
<tr>
<td>705129</td>
<td>CCOS</td>
<td>5</td>
<td>Driver for 9CCOS</td>
<td></td>
</tr>
<tr>
<td>705130</td>
<td>CDCOS</td>
<td>7</td>
<td>Driver for 9CDCOS</td>
<td></td>
</tr>
<tr>
<td>705131</td>
<td>CTANH</td>
<td>5</td>
<td>Driver for 9CTANH</td>
<td></td>
</tr>
<tr>
<td>705132</td>
<td>CDTANH</td>
<td>7</td>
<td>Driver for 9CDTANH</td>
<td></td>
</tr>
<tr>
<td>705133</td>
<td>CACOS</td>
<td>5</td>
<td>Driver for 9CACOS</td>
<td></td>
</tr>
<tr>
<td>705134</td>
<td>CDASIN</td>
<td>7</td>
<td>Driver for 9CDASIN</td>
<td></td>
</tr>
<tr>
<td>705135</td>
<td>CDACOS</td>
<td>7</td>
<td>Driver for 9CDACOS</td>
<td></td>
</tr>
<tr>
<td>705136</td>
<td>CLOG</td>
<td>5</td>
<td>Driver for 9CLOG</td>
<td></td>
</tr>
<tr>
<td>705137</td>
<td>CDLOG</td>
<td>7</td>
<td>Driver for 9CDLOG</td>
<td></td>
</tr>
<tr>
<td>705138</td>
<td>CEXP</td>
<td>5</td>
<td>Driver for 9CEXP</td>
<td></td>
</tr>
<tr>
<td>705139</td>
<td>CDEXP</td>
<td>7</td>
<td>Driver for 9CDEXP</td>
<td></td>
</tr>
<tr>
<td>705140</td>
<td>CSIN</td>
<td>5</td>
<td>Driver for 9CSIN</td>
<td></td>
</tr>
<tr>
<td>705141</td>
<td>CDSIN</td>
<td>7</td>
<td>Driver for 9CDSIN</td>
<td></td>
</tr>
<tr>
<td>705142</td>
<td>CTAN</td>
<td>5</td>
<td>Driver for 9CTAN</td>
<td></td>
</tr>
<tr>
<td>705143</td>
<td>CDTAN</td>
<td>7</td>
<td>Driver for 9CDTAN</td>
<td></td>
</tr>
<tr>
<td>705144</td>
<td>CATAN</td>
<td>5</td>
<td>Driver for 9CATAN</td>
<td></td>
</tr>
<tr>
<td>705145</td>
<td>CDATAN</td>
<td>7</td>
<td>Driver for 9CDATAN</td>
<td></td>
</tr>
<tr>
<td>705146</td>
<td>CSQRT</td>
<td>5</td>
<td>Driver for 9CSQRT</td>
<td></td>
</tr>
<tr>
<td>705147</td>
<td>CDSQRT</td>
<td>7</td>
<td>Driver for 9CDSQRT</td>
<td></td>
</tr>
<tr>
<td>705148</td>
<td>CASIN</td>
<td>5</td>
<td>Driver for 9CASIN</td>
<td></td>
</tr>
<tr>
<td>705149</td>
<td>CSINH</td>
<td>5</td>
<td>Driver for 9CSINH</td>
<td></td>
</tr>
<tr>
<td>705150</td>
<td>CCOSH</td>
<td>5</td>
<td>Driver for 9CCOSH</td>
<td></td>
</tr>
<tr>
<td>705151</td>
<td>CDSINH</td>
<td>7</td>
<td>Driver for 9CDSINH</td>
<td></td>
</tr>
<tr>
<td>705152</td>
<td>CDCOSH</td>
<td>7</td>
<td>Driver for 9CDCOSH</td>
<td></td>
</tr>
<tr>
<td>705153</td>
<td>FLOAT</td>
<td>5</td>
<td>Driver for 9ITOR</td>
<td></td>
</tr>
<tr>
<td>705154</td>
<td>DFLOAT</td>
<td>5</td>
<td>Driver for 9ITOD</td>
<td></td>
</tr>
<tr>
<td>705155</td>
<td>INT</td>
<td>5</td>
<td>Driver for 9RTOI</td>
<td></td>
</tr>
<tr>
<td>705156</td>
<td>IDINT</td>
<td>5</td>
<td>Driver for 9DINT</td>
<td></td>
</tr>
<tr>
<td>705157</td>
<td>SNGL</td>
<td>5</td>
<td>Driver for 9DINT</td>
<td></td>
</tr>
<tr>
<td>705158</td>
<td>CSNGL</td>
<td>5</td>
<td>Driver for 9DTOC</td>
<td></td>
</tr>
<tr>
<td>705159</td>
<td>CABS</td>
<td>5</td>
<td>Driver for 9CABS</td>
<td></td>
</tr>
<tr>
<td>705160</td>
<td>CDABS</td>
<td>7</td>
<td>Driver for 9CDABS</td>
<td></td>
</tr>
<tr>
<td>705161</td>
<td>ACOSF</td>
<td>5</td>
<td>Driver for 9ACOS</td>
<td></td>
</tr>
<tr>
<td>705162</td>
<td>ARCOS</td>
<td>5</td>
<td>Driver for 9ACOS</td>
<td></td>
</tr>
<tr>
<td>705163</td>
<td>ASINF</td>
<td>5</td>
<td>Driver for 9ASIN</td>
<td></td>
</tr>
<tr>
<td>Cat. No.</td>
<td>Name of Routine</td>
<td>Other Entries</td>
<td>Size</td>
<td>Description of Routine</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------</td>
<td>---------------</td>
<td>------</td>
<td>------------------------</td>
</tr>
<tr>
<td>705164</td>
<td>ARSIN</td>
<td></td>
<td>5</td>
<td>Driver for 9ASIN</td>
</tr>
<tr>
<td>705165</td>
<td>ATANF</td>
<td></td>
<td>11</td>
<td>Driver for 9ATAN1 and 9ATAN2</td>
</tr>
<tr>
<td>705166</td>
<td>COSH</td>
<td></td>
<td>5</td>
<td>Driver for 9COSH</td>
</tr>
<tr>
<td>705167</td>
<td>COSF</td>
<td></td>
<td>5</td>
<td>Driver for 9COS</td>
</tr>
<tr>
<td>705168</td>
<td>DARCOS</td>
<td></td>
<td>5</td>
<td>Driver for 9DARCSIN</td>
</tr>
<tr>
<td>705169</td>
<td>DARSIN</td>
<td></td>
<td>5</td>
<td>Driver for 9DASIN</td>
</tr>
<tr>
<td>705170</td>
<td>EXPF</td>
<td></td>
<td>5</td>
<td>Driver for 9EXP</td>
</tr>
<tr>
<td>705171</td>
<td>FLOATF</td>
<td></td>
<td>5</td>
<td>Driver for 9ITOR</td>
</tr>
<tr>
<td>705172</td>
<td>IFIX</td>
<td></td>
<td>5</td>
<td>Driver for 9RTOF</td>
</tr>
<tr>
<td>705173</td>
<td>LOG</td>
<td></td>
<td>5</td>
<td>Driver for 9ALOG</td>
</tr>
<tr>
<td>705174</td>
<td>LOG10</td>
<td></td>
<td>5</td>
<td>Driver for 9ALOG10</td>
</tr>
<tr>
<td>705175</td>
<td>SINF</td>
<td></td>
<td>5</td>
<td>Driver for 9SIN</td>
</tr>
<tr>
<td>705176</td>
<td>SINHF</td>
<td></td>
<td>5</td>
<td>Driver for 9SINH</td>
</tr>
<tr>
<td>705177</td>
<td>SQRF</td>
<td></td>
<td>5</td>
<td>Driver for 9SQR</td>
</tr>
<tr>
<td>705178</td>
<td>TANF</td>
<td></td>
<td>5</td>
<td>Driver for 9TAN</td>
</tr>
<tr>
<td>705179</td>
<td>TANHF</td>
<td></td>
<td>5</td>
<td>Driver for 9TANH</td>
</tr>
<tr>
<td>705180</td>
<td>ABS</td>
<td></td>
<td>4</td>
<td>Real absolute value</td>
</tr>
<tr>
<td>705181</td>
<td>AIMAG</td>
<td></td>
<td>4</td>
<td>Real imaginary part of complex</td>
</tr>
<tr>
<td>705182</td>
<td>AINT</td>
<td></td>
<td>6</td>
<td>Real integral value</td>
</tr>
<tr>
<td>705183</td>
<td>AMAX</td>
<td></td>
<td>16</td>
<td>Real maximum value</td>
</tr>
<tr>
<td>705184</td>
<td>AMAX1</td>
<td></td>
<td>16</td>
<td>Real maximum value</td>
</tr>
<tr>
<td>705185</td>
<td>AMAX0</td>
<td></td>
<td>17</td>
<td>Real maximum value of integer arguments</td>
</tr>
<tr>
<td>705186</td>
<td>AINN</td>
<td></td>
<td>16</td>
<td>Real minimum value</td>
</tr>
<tr>
<td>705187</td>
<td>AINN1</td>
<td></td>
<td>16</td>
<td>Real minimum value</td>
</tr>
<tr>
<td>705188</td>
<td>AMIN0</td>
<td></td>
<td>17</td>
<td>Real minimum value of integer arguments</td>
</tr>
<tr>
<td>705189</td>
<td>AMOD</td>
<td></td>
<td>11</td>
<td>Real remainder (modulo)</td>
</tr>
<tr>
<td>705190</td>
<td>CDUBLE</td>
<td></td>
<td>7</td>
<td>Complex to double complex conversion</td>
</tr>
<tr>
<td>705191</td>
<td>CDINT</td>
<td></td>
<td>12</td>
<td>Double complex integral value</td>
</tr>
<tr>
<td>705192</td>
<td>CINT</td>
<td></td>
<td>8</td>
<td>Complex integral value</td>
</tr>
<tr>
<td>705193</td>
<td>CMPLX</td>
<td></td>
<td>6</td>
<td>Complex from two real values</td>
</tr>
<tr>
<td>705194</td>
<td>CONJG</td>
<td></td>
<td>5</td>
<td>Complex conjugate</td>
</tr>
<tr>
<td>705195</td>
<td>DABS</td>
<td></td>
<td>4</td>
<td>Double precision absolute value</td>
</tr>
<tr>
<td>705196</td>
<td>DBLE</td>
<td></td>
<td>5</td>
<td>Real to double precision conversion</td>
</tr>
<tr>
<td>705197</td>
<td>DCMPLX</td>
<td></td>
<td>6</td>
<td>Double complex from two double precision values</td>
</tr>
<tr>
<td>705198</td>
<td>DCONJG</td>
<td></td>
<td>6</td>
<td>Double complex conjugate</td>
</tr>
<tr>
<td>705199</td>
<td>DDIM</td>
<td></td>
<td>9</td>
<td>Double precision positive difference</td>
</tr>
<tr>
<td>705200</td>
<td>DIM</td>
<td></td>
<td>8</td>
<td>Real positive difference</td>
</tr>
<tr>
<td>705201</td>
<td>DIMAG</td>
<td></td>
<td>5</td>
<td>Double precision imaginary part of double complex</td>
</tr>
<tr>
<td>705202</td>
<td>DINT</td>
<td></td>
<td>8</td>
<td>Double precision integral value</td>
</tr>
<tr>
<td>705203</td>
<td>DMAX</td>
<td></td>
<td>19</td>
<td>Double precision maximum value</td>
</tr>
<tr>
<td>705204</td>
<td>DMAX1</td>
<td></td>
<td>19</td>
<td>Double precision maximum value</td>
</tr>
<tr>
<td>705205</td>
<td>DMIN</td>
<td></td>
<td>19</td>
<td>Double precision minimum value</td>
</tr>
<tr>
<td>705206</td>
<td>DMIN1</td>
<td></td>
<td>19</td>
<td>Double precision minimum value</td>
</tr>
<tr>
<td>705207</td>
<td>DMOD</td>
<td></td>
<td>12</td>
<td>Double precision remainder (modulo)</td>
</tr>
<tr>
<td>705208</td>
<td>DREAL</td>
<td></td>
<td>4</td>
<td>Double precision real part of double complex</td>
</tr>
<tr>
<td>705209</td>
<td>DSIGN</td>
<td></td>
<td>8</td>
<td>Double precision first argument with sign of second argument</td>
</tr>
<tr>
<td>705210</td>
<td>IABS</td>
<td></td>
<td>4</td>
<td>Integer absolute value</td>
</tr>
<tr>
<td>705211</td>
<td>IAND</td>
<td></td>
<td>14</td>
<td>Integer Boolean product (AND)</td>
</tr>
<tr>
<td>705212</td>
<td>ICOMPL</td>
<td></td>
<td>5</td>
<td>Integer 1's complement (NOT)</td>
</tr>
<tr>
<td>705213</td>
<td>IDIM</td>
<td></td>
<td>8</td>
<td>Integer positive difference</td>
</tr>
<tr>
<td>705214</td>
<td>IEOR</td>
<td></td>
<td>14</td>
<td>Integer Boolean exclusive OR</td>
</tr>
<tr>
<td>705215</td>
<td>IEXCLR</td>
<td></td>
<td>14</td>
<td>Integer Boolean exclusive OR</td>
</tr>
<tr>
<td>705216</td>
<td>IF</td>
<td></td>
<td>17</td>
<td>Approximately equal or approximately zero</td>
</tr>
<tr>
<td>705217</td>
<td>INOT</td>
<td></td>
<td>5</td>
<td>Integer 1's complement (NOT)</td>
</tr>
<tr>
<td>Cat. No.</td>
<td>Name of Routine</td>
<td>Other Entries</td>
<td>Size</td>
<td>Description of Routine</td>
</tr>
<tr>
<td>----------</td>
<td>----------------</td>
<td>--------------</td>
<td>------</td>
<td>------------------------------------------------------------</td>
</tr>
<tr>
<td>705218</td>
<td>IOR</td>
<td></td>
<td>14</td>
<td>Integer Boolean sum (OR)</td>
</tr>
<tr>
<td>705219</td>
<td>ISIGN</td>
<td></td>
<td>8</td>
<td>Integer first argument with sign of second argument</td>
</tr>
<tr>
<td>705220</td>
<td>LOCF</td>
<td></td>
<td>5</td>
<td>Word address of argument</td>
</tr>
<tr>
<td>705221</td>
<td>MAX</td>
<td></td>
<td>16</td>
<td>Integer maximum value</td>
</tr>
<tr>
<td>705222</td>
<td>MAX0</td>
<td></td>
<td>16</td>
<td>Integer maximum value</td>
</tr>
<tr>
<td>705223</td>
<td>MAX1</td>
<td></td>
<td>17</td>
<td>Integer maximum value of real arguments</td>
</tr>
<tr>
<td>705224</td>
<td>MIN</td>
<td></td>
<td>16</td>
<td>Integer minimum value</td>
</tr>
<tr>
<td>705225</td>
<td>MIN0</td>
<td></td>
<td>16</td>
<td>Integer minimum value</td>
</tr>
<tr>
<td>705226</td>
<td>MIN1</td>
<td></td>
<td>17</td>
<td>Integer minimum value of real arguments</td>
</tr>
<tr>
<td>705227</td>
<td>MOD</td>
<td></td>
<td>8</td>
<td>Integer remainder (modulo)</td>
</tr>
<tr>
<td>705228</td>
<td>REAL</td>
<td></td>
<td>4</td>
<td>Real part of complex</td>
</tr>
<tr>
<td>705229</td>
<td>SIGN</td>
<td></td>
<td>8</td>
<td>Real first argument with sign of second argument</td>
</tr>
<tr>
<td>705230</td>
<td>ABSF</td>
<td></td>
<td>4</td>
<td>Absolute value</td>
</tr>
<tr>
<td>705231</td>
<td>DIMF</td>
<td></td>
<td>8</td>
<td>Positive difference</td>
</tr>
<tr>
<td>705232</td>
<td>SIGNF</td>
<td></td>
<td>8</td>
<td>Real first argument with sign of second argument</td>
</tr>
<tr>
<td>705233</td>
<td>SSWITCH</td>
<td></td>
<td>17</td>
<td>Test sense switch</td>
</tr>
<tr>
<td>705234</td>
<td>SLITET</td>
<td></td>
<td>17</td>
<td>Test sense light</td>
</tr>
<tr>
<td>705235</td>
<td>SLITE</td>
<td></td>
<td>5</td>
<td>Set sense light</td>
</tr>
<tr>
<td>705236</td>
<td>OVERFL</td>
<td></td>
<td>8</td>
<td>Test for floating overflow</td>
</tr>
<tr>
<td>705237</td>
<td>DVCHK</td>
<td></td>
<td>8</td>
<td>Test for floating overflow</td>
</tr>
<tr>
<td>705238</td>
<td>EXIT</td>
<td></td>
<td>9</td>
<td>Exit to the Monitor</td>
</tr>
<tr>
<td>705239</td>
<td>EOFSET</td>
<td></td>
<td>43</td>
<td>Set up end-of-file exit</td>
</tr>
<tr>
<td>705240</td>
<td>SETEOF</td>
<td></td>
<td>41</td>
<td>Set up end-of-file exit</td>
</tr>
<tr>
<td>705241</td>
<td>BUFFERIN</td>
<td></td>
<td>52</td>
<td>Direct input</td>
</tr>
<tr>
<td>705242</td>
<td>BUFFEROU</td>
<td></td>
<td>43</td>
<td>Direct output</td>
</tr>
<tr>
<td>705243</td>
<td>ABORTSET</td>
<td></td>
<td>21</td>
<td>Set up abort exit and severity level</td>
</tr>
</tbody>
</table>
2. PROGRAM DESCRIPTIONS

The following program descriptions of the routines in the 5/7 FORTRAN IV Library are in sequence by SDS Software Library Catalog Number.
705001 9ALOG (7ALOG) (7ALOG1) (7ALOG2), REAL NATURAL LOGARITHM

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, 10, and 11.

Purpose: Calculates \( \ln(X) \),

where 

\[ X = \text{FORTRAN IV REAL entity}. \]

Size: 79

Subroutines Used: 8TO (051), 9ERROR (065), BT1 (091)

(Indirectly): 7ERROR (066), BTINIT (092), BTERROR (093)

705002 9DLOG (7DLOG), (7DLOG1), (7DLOG2), DOUBLE PRECISION NATURAL LOGARITHM

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4, as well as six words in the stack whose stack pointer doubleword location is in register 6.

Purpose: Calculates \( \ln(X) \),

where 

\[ X = \text{FORTRAN IV DOUBLE PRECISION entity}. \]

Size: 116

Subroutines Used: 8TO (051), 9ERROR (065)

(Indirectly): 7ERROR (066), BTINIT (092), BTERROR (093)

705003 9EXP (7EXP1) (7EXP2), REAL EXPONENTIAL

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, and 11 (reentrant version also uses register 10).

Purpose: Calculates \( e^X \),

where 

\[ X = \text{FORTRAN IV REAL entity}. \]

Size: 59

Subroutines Used: 8TO (051), 9ERROR (065)

(Indirectly): 7ERROR (066), BTINIT (092), BTERROR (093)
705004  9DEXP (7DEXP1) (7DEXP2), DOUBLE PRECISION EXPONENTIAL

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4, as well as six words in the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates $e^X$

where

$X$ = FORTRAN IV DOUBLE PRECISION entity.

Size: 87

Subroutines Used: 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705005  9SIN(9COS), REAL SINE AND COSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2 and 9. The reentrant version also uses register 11.

Purpose: Calculates sin(X) or cos(X),

where

$X$ = FORTRAN IV REAL entity.

Size: 90

Subroutines Used: 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705006  9DSIN (9DCOS), DOUBLE PRECISION SINE AND COSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, and 11. The reentrant version also uses registers 12 and 13.

Purpose: Calculates sin(X) or cos(X),

where

$X$ = FORTRAN IV DOUBLE PRECISION entity.

Size: 133

Subroutines Used: 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705007  9ATAN1 (9ATAN2), REAL ARCTANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: Y in register 8, X in register 9 (if 9ATAN2), register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2 and 10 (reentrant version also uses register 11).

Purpose: Calculates $\tan^{-1}(Y)$ or $\tan^{-1}(Y/X)$,

where

$X$ and $Y$ = FORTRAN IV REAL entities.

Size: 91

Subroutines Used: 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)
705008 9DATAN1 (9DATAN2), DOUBLE PRECISION ARCTANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequences: Y in register pair B-9, X in register pair 10-11 (if 9DATAN2), register 6 is link, result returned in register pair B-9. Also uses (and does not preserve) registers 2 and 12 (reentrant version also uses register 13).

Purpose: Calculates \( \tan^{-1}(Y) \) or \( \tan^{-1}(Y/X) \), where

\[ X \text{ and } Y \text{ FORTRAN IV DOUBLE PRECISION entities.} \]

Size: 121

Subroutines Used: 8T0 (051), 9ERROR (065), 8T1 (091)
(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705009 9SQRT, REAL SQUARE ROOT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register B. Also uses (and does not preserve) registers 2, 9, and 10 (reentrant version also uses register 11).

Purpose: Calculates \( \sqrt{X} \), where

\[ X \text{ FORTRAN IV REAL entity.} \]

Size: 38

Subroutines Used: 8T0 (051), 9ERROR (065), 8T1 (091)
(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705010 9DSQRT, DOUBLE PRECISION SQUARE ROOT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair B-9, register 6 is link, result returned in register pair B-9. Also uses (and does not preserve) registers 2 and 10 (reentrant version also uses registers 11, 12, and 13).

Purpose: Calculates \( \sqrt{X} \), where

\[ X \text{ FORTRAN IV DOUBLE PRECISION entity.} \]

Size: 63

Subroutines Used: 8T0 (051), 9ERROR (065), 8T1 (091)
(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705011 9SINH (9COSH), REAL HYPERBOLIC SINE/COSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register B. Also uses (and does not preserve) registers 2, 9, 10, and 11.

Purpose: Calculates sinh (X) or cosh (X), where

\[ X \text{ FORTRAN IV REAL entity.} \]

Size: 78

Subroutines Used: 9EXP (003), 8T0 (051), 9ERROR (065), 8T1 (091)
(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)
705012  9DSINH (9DCOSH), DOUBLE PRECISION HYPERBOLIC SINE/COSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, and 11. Reentrant version also uses registers 12 and 13.

Purpose: Calculates sinh(X) or cosh(X),
where
  X = FORTRAN IV DOUBLE PRECISION entity.

Size: 81

Subroutines Used: 9DEXP (004), 9ERROR (065), BT1 (091)
(Indirectly): BT0 (051), 7ERROR (066), BTINIT (092), 8ERROR (093)

705013  9TANH, REAL HYPERBOLIC TANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, and 10. The reentrant version also uses registers 4 and 11 plus two words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates tanh(X),
where
  X = FORTRAN IV DOUBLE PRECISION entity.

Size: 68

Subroutines Used: 9DEXP (004), BT1 (091)
(Indirectly): BT0 (051), 7ERROR (066), 7ERROR (066), BTINIT (092), BTERROR (093)

705015  9ASIN (9ACOS), REAL ARCSINE/ARCCOSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, and 10. The reentrant version also uses registers 4 and 11 plus two words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates sin^{-1}(X) or cos^{-1}(X),
where
  X = FORTRAN IV REAL entity.

Size: 26

Subroutines Used: 9ATAN1 (007), 9SQRT (009), 9ERROR (065), BT1 (091)
(Indirectly): BT0 (051), 7ERROR (066), BTINIT (092), BTERROR (093)
9DASIN (9DACOS), DOUBLE PRECISION ARCSINE/ARCCOSINE

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and five words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( \sin^{-1}(X) \) or \( \cos^{-1}(X) \), where

\( X = \text{FORTRAN IV DOUBLE PRECISION entity.} \)

Size: 65

Subroutines Used: 9DATANI (008), 9DSQRT (010), 9ERROR (065), 8T1 (091)

(Indirectly): 8T0 (051), 7ERROR (066), BTINIT (092), BERROR (093)

9DTAN, DOUBLE PRECISION TANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and six words in the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( \tan(X) \), where

\( X = \text{FORTRAN IV DOUBLE PRECISION entity.} \)

Size: 110

Subroutines Used: 8T0 (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), BTINIT (092), BERROR (093)

9TAN, REAL TANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2 and 9. Reentrant version also uses registers 10 and 11.

Purpose: Calculates \( \tan(X) \), where

\( X = \text{FORTRAN IV REAL entity.} \)

Size: 70

Subroutines Used: 8T0 (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), BTINIT (092), BERROR (093)

9ALOG10, REAL COMMON LOGARITHM

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, 10, and 11.

Purpose: Calculates \( \log_{10}(X) \), where

\( X = \text{FORTRAN IV REAL entity.} \)

Size: 19

Subroutines Used: 9ALOG (001), 9ERROR (065), 8T1 (091)

(Indirectly): 8T0 (051), 7ERROR (066), BTINIT (092), BERROR (093)
705020 9DLOG10, DOUBLE PRECISION COMMON LOGARITHM

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and seven words in the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( \log_{10}(X) \),

where

\[ X = \text{FORTRAN IV DOUBLE PRECISION entity.} \]

Size: 21

Subroutines Used: 9DLOG (002), 9ERROR (066), 8T1 (091)

(Indirectly): 8TO (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705021 9PWRII, INTEGER RAISED TO INTEGER POWER

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: I in register 9, J in register 11, register 6 is link, result returned in register 9. Also uses (and does not preserve) registers 2, 10, and 11.

Purpose: Calculates \( I^J \),

where

\[ I, J = \text{FORTRAN IV INTEGER entities.} \]

Size: 37

Subroutines Used: 9ERROR (066)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705022 9PWRRI, REAL RAISED TO INTEGER POWER

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register 8, I in register 9, J in register 11, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 10, and 11.

Purpose: Calculates \( X^I \),

where

\[ X = \text{FORTRAN IV REAL entity} \]
\[ I = \text{FORTRAN IV INTEGER entity} \]

Size: 35

Subroutines Used: 9ERROR (066)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

705023 9PWRRI2, DOUBLE PRECISION RAISED TO INTEGER POWER

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: X in register pair 8-9, I in register 11, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, and 13.

Purpose: Calculates \( X^I \),

where

\[ X = \text{FORTRAN IV DOUBLE PRECISION entity} \]
\[ I = \text{FORTRAN IV INTEGER entity} \]

Size: 35

Subroutines Used: 9ERROR (066)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)
**9PWRCI, COMPLEX RAISED TO INTEGER POWER**

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: Z in register pair 8-9, I in register 11, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses register 4 and four words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculate \( Z^{**I} \),

where

\[ \begin{align*} Z &= \text{FORTRAN IV COMPLEX entity} \\ I &= \text{FORTRAN IV INTEGER entity} \end{align*} \]

Size: 67

Subroutines Used: 9PWRI (022), 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

---

**9PWRLD, DOUBLE COMPLEX RAISED TO INTEGER POWER**

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: Z in register pair 8-9, I in register 13, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) register 2. The reentrant version also uses register 4 and nine words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( Z^{**I} \),

where

\[ \begin{align*} Z &= \text{FORTRAN IV DOUBLE COMPLEX entity} \\ I &= \text{FORTRAN IV INTEGER entity} \end{align*} \]

Size: 74

Subroutines Used: 9PWRI (022), 8TO (051), 9ERROR (065), 8T1 (091)

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (093)

---

**9PWRRI, REAL RAISED TO REAL POWER**

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: X in register 8, Y in register 10, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 9, and 11. The reentrant version also uses register 4 and two words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( X^{**Y} \),

where

\[ \begin{align*} X\text{ and } Y &= \text{FORTRAN IV REAL entities} \end{align*} \]

Size: 57

Subroutines Used: 9ALOG (001), 9EXP (003), 9ERROR (065), 8T1 (091)

(Indirectly): 8TO (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

---

**9PWRRD, DOUBLE PRECISION RAISED TO DOUBLE PRECISION POWER**

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: X in register pair 8-9, Y in register pair 10-11, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses register 4 and 18 words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( X^{**Y} \),

where

\[ \begin{align*} X\text{ and } Y &= \text{FORTRAN IV DOUBLE PRECISION entities} \end{align*} \]

Size: 57

Subroutines Used: 9DLOG (002), 9DEXP (004), 9ERROR (065), 8T1 (091)

(Indirectly): 8TO (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)
705028 9PWRC, COMPLEX RAISED TO COMPLEX POWER

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pair 8-9, W in register pair 10-11, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) register 2. The reentrant version also uses register 4 and six words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates $Z^W$, where $Z$ and $W$ = FORTRAN IV COMPLEX entities.

Size: 40

Subroutines Used: 9CLOG (036), 9CEXP (038), 9ERROR (065), 8T1 (091)

(Indirectly): 9ALOG (001), 9EXP (003), 9ATAN1 (007), 8T0 (051), 7SIN (052), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705029 9PWKK, DOUBLE COMPLEX RAISED TO DOUBLE COMPLEX POWER

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, W in register pairs 12-13 and 14-15, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) register 2. The reentrant version also uses register 4 and sixteen words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates $Z^W$, where $Z$ and $W$ = FORTRAN IV DOUBLE COMPLEX entities.

Size: 41

Subroutines Used: 9CLOG (037), 9CEXP (039), 9ERROR (065), 8T1 (091)

(Indirectly): 9DLOG (002), 9DEXP (004), 9DSIN (006), 9DATAN1 (008), 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705030 9CADD (9CSUB), COMPLEX ADD/SUBTRACT

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: $Z_1$ in register pair 8-9, $Z_2$ in register pair 10-11, register 6 is link, result returned in register pair 8-9.

Purpose: Calculates $Z_1 + Z_2$ or $Z_1 - Z_2$.

where

$Z_1$ and $Z_2$ = FORTRAN IV COMPLEX entities.

Size: 6

Subroutines Used: (none)

(Indirectly): (none)

705031 9CMUL, COMPLEX MULTIPLY

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: $Z_1$ in register pair 8-9, $Z_2$ in register pair 10-11, register 6 is link, result returned in register pair 8-9. The reentrant version also uses register 4 and four words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates $Z_1 \cdot Z_2$.

where

$Z_1$ and $Z_2$ = FORTRAN IV COMPLEX entities.

Size: 10

Subroutines Used: 8T0 (051), BT1 (091)

(Indirectly): (none)
705032 9CDIV, COMPLEX DIVIDE

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z1 in register pair 8-9, Z2 in register pairs 10-11, register 6 is link, result returned in register pair 8-9. The reentrant version also uses register 4 and four words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates Z1/Z2',
where
Z1 and Z2 = FORTRAN IV COMPLEX entities.

Size: 19
Subroutines Used: 8TO (051), 8TI (091)
(Indirectly): (none)

705033 9KADD (9KSUB), DOUBLE COMPLEX ADD/SUBTRACT

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z1 in register pairs 8-9 and 10-11, Z2 in register pairs 12-13 and 14-15, register 6 is link, result returned in register pairs 8-9 and 10-11.

Purpose: Calculates Z1 + Z2 or Z1 - Z2',
where
Z1 and Z2 = FORTRAN IV DOUBLE COMPLEX entities.

Size: 6
Subroutines Used: (none)
(Indirectly): (none)

705034 9KMUL, DOUBLE COMPLEX MULTIPLY

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z1 in register pairs 8-9 and 10-11, Z2 in register pairs 12-13 and 14-15, register 6 is link, result returned in register pairs 8-9 and 10-11. The reentrant version also uses register 4 and nine words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates Z1 * Z2',
where
Z1 and Z2 = FORTRAN IV DOUBLE COMPLEX entities.

Size: 10
Subroutines Used: 8TO (051), 8TI (091)
(Indirectly): (none)

705035 9KDIV, DOUBLE COMPLEX DIVIDE

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z1 in register pairs 8-9 and 10-11, Z2 in register pairs 12-13 and 14-15, register 6 is link, result returned in register pairs 8-9 and 10-11. The reentrant version also uses register 4 and nine words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates Z1/Z2',
where
Z1 and Z2 = FORTRAN IV DOUBLE COMPLEX entities.

Size: 17
Subroutines Used: 8TO (051), 8TI (091)
(Indirectly): (none)
705036 9CLOG (7CLOG), COMPLEX NATURAL LOGARITHM

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and two words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates ln(Z),
where
\[ Z \text{ = FORTRAN IV COMPLEX entity.} \]

Size: 56

Subroutines Used: 9ALOG (001), 9ATAN1 (007), 9ERROR (065), BT1 (091)

(Indirectly): BT0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705037 9CDLOG (7CDLOG), DOUBLE COMPLEX NATURAL LOGARITHM

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) register 2. The reentrant version also uses register 4 and eight words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates ln(Z),
where
\[ Z \text{ = FORTRAN IV DOUBLE COMPLEX entity.} \]

Size: 53

Subroutines Used: 9DLOG (002), 9DATAN1 (008), 9ERROR (066), BT1 (091)

(Indirectly): BT0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705038 9CEXP, (7CEXP), COMPLEX EXPONENTIAL

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and four words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( e^Z \),
where
\[ Z \text{ = FORTRAN IV COMPLEX entity.} \]

Size: 20

Subroutines Used: 7SIN (052), 9ERROR (065), BT1 (091)

(Indirectly): 9EXP (003), BT0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705039 9CDEXP (7CDEXP), DOUBLE COMPLEX EXPONENTIAL

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses register 4 and eight words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates \( e^Z \),
where
\[ Z \text{ = FORTRAN IV DOUBLE COMPLEX entity.} \]

Size: 28

Subroutines Used: 9DEXP (004), 9DSIN (006), 9ERROR (065), BT1 (091)

(Indirectly): BT0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)
Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pair 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and five words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates sin(Z), cos(Z), sinh(Z), or cosh(Z), where

\[ Z = \text{FORTRAN IV COMPLEX entity.} \]

Size: 86

Subroutines Used: 8TO (051), 7SIN (052), 9ERROR (065), 8T1 (091)

(Indirectly): 9EXP (003), 7ERROR (066), 8TINIT (092), 8TERROR (093)

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses registers 4 and eighteen words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates tan(Z) or tanh(Z), where

\[ Z = \text{FORTRAN IV DOUBLE COMPLEX entity.} \]

Size: 112

Subroutines Used: 9EXP (004), 9DSIN (006), 9ERROR (065), 8T1 (091)

(Indirectly): 8TO (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)
9CATAN, COMPLEX ARCTANGENT

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9, register 6 is link, result returned in register pair 8-9. Also uses (and does not preserve) registers 2, 10, 11, 12, and 13. The reentrant version also uses register 4 and three words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates tan⁻¹(Z),
where
   Z = FORTRAN IV COMPLEX entity.

Size: 76

Subroutines Used: 9ALOG (001), 9ATAN1 (007), 9ERROR (065), 8T1 (091)
(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

9CDATAN, DOUBLE COMPLEX ARCTANGENT

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, register 6 is link, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses register 4 and sixteen words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates tan⁻¹(Z),
where
   Z = FORTRAN IV DOUBLE COMPLEX entity.

Size: 104

Subroutines Used: 9DLOG (002), 9DATAN1 (008), 9ERROR (065), 8T1 (091)
(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

9CSQRT, (9CABS), COMPLEX SQUARE ROOT AND MODULUS

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pair 8-9, register 6 is link, result returned in register pair 8-9 (register 8 if 9CABS). Also uses (and does not preserve) registers 2, 10, and 11. The reentrant version also uses registers 4, 12, and 13.

Purpose: Calculates √Z or |Z|,
where
   Z = FORTRAN IV COMPLEX entity.

Size: 88

Subroutines Used: 9SQRT (009), 9ERROR (065), 8T1 (091)
(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

9CDSQRT, (9CDABS), (9CDRT), DOUBLE COMPLEX SQUARE ROOT AND MODULUS

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z in register pairs 8-9 and 10-11, register 6 is link, result returned in register pairs 8-9 and 10-11 (only 8-9 if 9CDABS). Also uses (and does not preserve) registers 2, 12, and 13. The reentrant version also uses register 4 and eight words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates √Z or |Z|,
where
   Z = FORTRAN IV DOUBLE COMPLEX entity.

Size: 92

Subroutines Used: 9DSQRT (010), 9ERROR (065), 8T1 (091)
(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)
Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: register 6 is link, (for complex) Z in register pair 8-9, result returned in register pair 8-9; (for double complex) Z in register pair 8-9 and 10-11, result returned in register pairs 8-9 and 10-11. Also uses (and does not preserve) registers 2 and 13. The reentrant version also uses registers 4 and 12 plus 24 words from the stack whose stack pointer doubleword location is in register 0.

Purpose: Calculates sin^{-1}(Z) or cos^{-1}(Z),

where

Z = FORTRAN IV COMPLEX entity,

or

Z = FORTRAN IV DOUBLE COMPLEX entity.

Size: 91

Subroutines Used: 9CDLOG (037), 9CDSQRT (047), 8TO (091)

(Indirectly): 9DLOG (002), 9DATAN (008), 9DSQRT (010), 8TO (051), 9ERROR (065), 7ERROR (066), 8TINIT (092), 8TERROR (093)

Purpose: Provides 17 words of external temps that may be used by the library. Any routine may use these temps but should not normally assume that they will be preserved by other routines, unless this is carefully planned as has been done in the math routines.

Size: 17

Subroutines Used: (none)

(Indirectly): (none)

Purpose: Calculates e^X, sin(Y), and cos(Y),

where

X and Y = FORTRAN IV REAL entities.

Used only by 9CEXP and 9CSIN.

Size: 85

Subroutines Used: 9EXP (003), 8TO (051), 8TO (091)

(Indirectly): 9ERROR (065), 7ERROR (066), 8TINIT (092), 8TERROR (093)
705053  9IFR, REAL APPROXIMATE EQUALITY TEST

Calling Sequence: (ARG0) = ARG1
(ARG1) = ARG2
(ARG2) = EPSILON
BAL, LL  9IFR

Purpose: Accepts three real arguments and returns an integer result in AL.
When neither ARG1 nor ARG2 is zero, the result is zero if
ABS(ARG1-ARG2) • LE EPSILON*AMIN(ABS(ARG1)-ABS(ARG2))
otherwise, the result is a random integer with the sign of (ARG1-ARG2).
If either ARG1 or ARG2 is zero, the result is zero if
ABS(ARG1+ARG2) • LE EPSILON
otherwise, the result has the sign of (ARG1-ARG2).
Size: 21
Subroutines Used: (none)
(Indirectly): (none)

705055  9ITOD (9ITOR), INTEGER TO FLOATING CONVERSIONS

Calling Sequence: (AI) = integer
BAL, LL  9ITOD
(AI) = integer
BAL, LL  9ITOR

Purpose: Converts integer in AI to double precision in AD. This automatically produces a real result in AR, except that values greater than 2**21 may not be rounded correctly.
Size: 7
Subroutines Used: (none)
(Indirectly): (none)

705056  9DTOI (9RTOI), FLOATING TO INTEGER CONVERSIONS

Calling Sequence: (AI) = real value
BAL, LL  9DTOI
(AR) = real value
BAL, LL  9RTOI

Purpose: Converts double precision value in AD to an integer in AI. Real values are converted to double precision first. Arguments whose integer magnitude is out of range may produce meaningless results.
Size: 8
Subroutines Used: (none)
(Indirectly): (none)

705057  9DTOR, DOUBLE PRECISION TO REAL CONVERSION

Calling Sequence: (AD) = double precision value
BAL, LL  9DTOR

Purpose: Rounds the double precision number before taking the single precision part, thereby making the real value accurate to within half a bit. No unnormalized results are produced (as would be the case if the second word were just discarded). Values too large to represent in single precision cause an overflow, after which the Trap routine returns the maximum value within machine range. Exits with (AR) = real value.
Size: 24
Subroutines Used: (none)
(Indirectly): (none)
9KTOC, DOUBLE COMPLEX TO COMPLEX CONVERSION

Calling Sequence: (AK) = double complex value
BAL, LL 9KTOC

Purpose: Converts double complex to complex value. Exits with (AC) = complex value.

Size: 8
Subroutines Used: 8TO (051), 9DTOR (057)
(Indirectly): 8T1 (091)

9SETUPO, SET UP ZERO ARGUMENTS

Calling Sequence: BAL, LR 9SETUPO

Purpose: Sets zero arguments in a standard receiving sequence. Essentially, this involves doing nothing. However, 9SETUPO increments LC by the contents of NA (which should be zero) as error recovery in case the call contains more than zero arguments.

Size: 2
Subroutines Used: (none)
(Indirectly): (none)

9SETUP1, SET UP ONE ARGUMENT

Calling Sequence: BAL, LR 9SETUP1

Purpose: Sets one argument in a standard receiving sequence. The calling sequence word may be indirect. See also 9SETUPN (705062).

Size: 7
Subroutines Used: (none)
(Indirectly): (none)

9SETUP2, SET UP TWO ARGUMENTS

Calling Sequence: BAL, LR 9SETUP2
type, P dummy temp 1
type, P dummy temp 2

Purpose: Sets up two arguments in a standard receiving sequence. The calling sequence words may be indirect. See also 9SETUPN (705062).

Size: 13
Subroutines Used: (none)
(Indirectly): (none)
### Calling Sequence

For **9SETUPN**:  
\[(ND) = \text{number of dummies}\]  
\[\text{BAL, LR} \rightarrow 9SETUPN\]  
\[\text{type, P} \rightarrow \text{dummy temp 1}\]  
\[\text{type, P} \rightarrow \text{dummy temp 2}\]  
\[\vdots\]  
\[\text{type, P} \rightarrow \text{dummy temp ND}\]

For **9SETUPM**:  
\[(MNA) = \text{minimum number of arguments}\]  
\[(MXA) = \text{maximum number of arguments}\]  
\[\text{BAL, LR} \rightarrow 9SETUPM\]  
\[\text{type, P} \rightarrow \text{dummy temp 1}\]  
\[\text{type, P} \rightarrow \text{dummy temp 2}\]  
\[\vdots\]  
\[\text{type, P} \rightarrow \text{dummy temp MXA}\]

where

- **type** = a mask of permissible types
- **P** (Protected) = 8 (if argument is to be stored into)  
  = 0 (if argument is not to be stored into)

### Purpose

- **9SETUPN**: Sets up any fixed number of arguments in a standard receiving sequence.  
  (For zero, one, or two arguments it is faster to use 9SETUPO, 9SETUP1, or 9SETUP2, respectively.)
- **9SETUPM**: Sets up a variable number of arguments within a fixed range.
- **7SET**: Special entry used by 9SETUPV.

These routines return to location (LR) + (ND) after setting up GRUNCH (the subprogram that called 9SETUPN (9SETUPM)) to return to (LC) + (NA). Remote calling sequences are not handled.

### Error Recovery

When the number of calling arguments disagrees with the number of receiving dummies and there are too many arguments (too few dummies), the first n arguments are used; if there are too few arguments (too many dummies), the first n dummies are set up and the other dummies remain the same as on the previous call to GRUNCH. 9SETUPN can accept calling sequences that are too short, thereby enabling it to work as 9SETUPM in the non-debug mode. The minimum acceptable number of arguments is passed in register 4 and is ignored and preserved.

### Size

- **9SETUPN**: 14
- **9SETUPM**: 42

### Subroutines Used

- **9SETUPN**: (none)
- **9SETUPM**: (none)

### Notes

- Subroutine Used: (none)
- Indirectly Used: (none)
**70063 9SETUPV, SET UP A VARIABLE NUMBER OF ARGUMENTS (MULTIPLE DUMMY)**

**Calling Sequence:**

\[
\text{BAL, LR 9SETUPV}
\]

\[
\text{type, P fixed dummy temp 1}
\]

\[
\text{type, P fixed dummy temp 2}
\]

\[
\text{...}
\]

\[
\text{type, P fixed dummy temp (number of dummies (negative))}
\]

\[
\text{type, P multiple dummy temp}
\]

where

- `type` = a mask of permissible types
- `P` (Protected) = 8 (if argument is to be stored into)
- 0 (if argument is not to be stored into)

Neither of these conditions is checked in non-debug mode.

It is assumed that the following parameters have been set up by the program that called the subprogram using the setup routine:

- `(NA)` = number of arguments
- `(LC)` = location of first argument word

After setting up the multiple dummy, 9SETUPV branches to 7SET (an entry in 9SETUPN) to set up the fixed dummies. 7SET returns directly to GRUNCH (the subprogram that called 9SETUPV) at location `(LC) - (ND)` after setting up GRUNCH to return to `(LC) + (NA)`. Remote calling sequences are not handled.

**Purpose:** Sets up a variable number of arguments in a standard receiving sequence (involving a multiple dummy).

**Error Recovery:** There can never be too many arguments. Too few arguments mean there are not enough to satisfy all the fixed dummies, in which case the return for both 9SETUPV and GRUNCH will still be to the correct place. Argument count for the multiple dummy (register 4) will be a negative number indicating the discrepancy of the calling sequence (-2 = two arguments short). The multiple dummy pointer will point beyond the calling sequence and should not be used (just as it should not be used if the number of arguments for the multiple dummy is zero). Only the first fixed dummy will be set up; others remain the same as on the previous call to GRUNCH.

**Size:** 11

**Subroutines Used:** 9SETUPN (062)

(Indirectly): (none)

---

**70064 9INITIAL, RUN-TIME INITIALIZATION**

**Calling Sequence:**

\[
\text{BAL, LL 9INITIAL}
\]

**Purpose:** This program must be called immediately at the beginning of every FORTRAN execution, i.e., at the start of the main program. FORTRAN IV object programs (with appropriate library routines) do not use post-initialization or initialization dependent on loading. All necessary initialization is done executable in 9INITIAL, thereby enabling an object program to be restarted at any time, whether it has finished or aborted or is still running.

This initialization includes turning off all the sense lights and the floating overflow trigger, resetting the end-of-file and abort exits to go to the Monitor, setting the abort severity to 8, and informing the I/O package that there is no I/O in progress.

9INITIAL also sets up the floating control and the traps. Floating overflow traps are directed to a trap handler within 9INITIAL. This trap handler sets the FORTRAN floating overflow trigger when an overflow occurs and then returns a maximum value default result.

**Size:** 47

**Subroutines Used:** 8TINIT (092)

(Indirectly): (none)
Calling Sequence:  
(LL)  = exit from math routine
BAL, LE  = ERROR
GEN, 16, 4, 1, 1, 10  = N, SEV, D, C, DEF
TEXT  = 'MATHNAME'

where

N = code number for first part of message
SEV = error severity
D = 1 (if precision is double)
C = 1 (if complex or double complex)
DEF = code number for default result and second part of message

Note: 9ERROR returns directly to the caller of the math routine. It uses LE only to locate arguments.

Purpose: When improper arguments cause overflow, loss of significance, or undefined results in a math routine, the math routine calls 9ERROR to

1. Prepare a default result in the appropriate register.
2. Construct an error message explaining both the cause of the error and the recovery.
3. Call 7ERROR, which prints the message and then decides, on the basis of the severity, whether to abort or return to the math routine user with the default result.

The parameter N is used to select the first part of the error message that will be printed, as follows:

N = 1  zero or negative argument
    2  magnitude of argument too large
    3  zero arguments
    4  negative argument
    5  zero to non-positive power
    6  argument too large
    7  zero argument
    8  singularity at + or -i
    9  negative to non-integral power

The parameter DEF determines the rest of the message and the default result to be returned. It is interpreted as

DEF = 0  no significance; result = zero
      1  overflow; result = maximum. (floating)
      2  overflow; result = maximum negative. (floating)
      4  overflow; result = maximum. (integer)

The severity code (SEV) is used only by 7ERROR in determining whether to abort or return.

Size: 110

Subroutines Used: 7ERROR (066), 8ERROR (093)
(Indirectly): 8TINIT (092)
Colling Sequence:

For 7ERROR:

EL = entry location
EN = address of the two-word entry name
BL = BA (TEXTIC 'ERROR MESSAGE')
ES = error severity level
BAL, LE = 7ERROR

For 7ERRHEAD:

EL = entry location
EN = address of first word of two-word error name
BAL, LE = 7ERRHEAD

For 7ERRTEXT:

BL = BA (TEXTIC 'ERROR MESSAGE')
ES = error severity level
BAL, LE = 7ERRTEXT

For 7ERRMARK:

BB = BA (beginning of buffer)
BP = BA (mark character)
BE = BA (end of buffer)
BAL, LE = 7ERRMARK

For 7ERRINIT:

BAL, LE = 7ERRINIT

For 7PRC:

CH = character
BAL, LE = 7PRC

For 7PRQ:

BL = BA (TEXTIC 'QUOTE STRING')
BAL, LE = 7PRQ

For 7PAC:

BL = BA (first character of the string)
NC = number of characters in the string
BAL, LE = 7PAC

For 7PHC:

AH = hexadecimal value
NC = number of hexadecimal digits
BAL, LE = 7PHC

For 7PRL:

BAL, LE = 7PRL

For 7BUFOUT:

BL = BA (first character of the string)
NC = number of characters in the string
BAL, LE = 7BUFOUT

For 7BUFOUTC:

BL = BA (TEXTIC 'QUOTE STRING')
BAL, LE = 7BUFOUTC

Purpose:

7ERROR: Prints error heading and message.
7ERRHEAD: Prints error heading of two lines: a blank line and a line pro-
claiming a FORTRAN run-time error.

This program is the universal error handler for all FORTRAN run-time errors. It does not have any control over Monitor error conditions, but all FORTRAN error conditions (including those in the math routines, which first go to 9ERROR) must come through 7ERROR. Most of them do so by simply calling 7ERROR and providing it with all of the pertinent information, which includes the error message location and severity level, the name of the routine in which the error occurred, and the location at which that routine was called.

When an error message must be constructed, or more than one line is to be printed (e.g., FORMAT errors), it is necessary to call 7ERROR in parts. A call on 7ERROR is equivalent to calls on 7ERRHEAD and 7ERRTEXT, in that order. 7ERRHEAD prints the heading

FORTRAN RUN-TIME ERROR IN 'name', CALLED AT LOC X'xxxxx'.

7ERRTEXT prints a message and a blank line, and then determines whether or not to abort.

In determining whether or not to abort, the severity of the current error, which is passed in register ES, is compared with the value in 8ABRTSEV: if it is greater or equal, an abort is performed; otherwise, a return is made to the caller of 7ERROR. The standard abort severity (in 8ABRTSEV) is initialized (by 9INITIAL) to a value of 8, so that the typical "warning level" severities of 4 and 7 will not abort (since a recovery can be made), while level 15 will abort.
If it is desired to abort on less serious errors, 8ABRTSEV can be changed by use of the
ABORTSET subroutine (705243). Note that ABORTSET can also be used to specify an abort
exit other than the Monitor. The user can provide a location to which a transfer will be
made when an abort level error occurs, but he is then obliged to determine how to continue
his job.

The other entries to 7ERROR are used primarily for building up error messages (usually in
the error message buffer (8MSGBUF)), to be output either by 7ERRTEXT (for the last line of
the message) or 7BUFOUT/7BUFOUTC (for any preceding lines in a multiline message). Note
that five of the entries have the same names as corresponding POPs in the compiler.

Size: 144
Subroutines Used: 8TINIT (092), 8TERROR (093)

9BCDREAD (9READ), BCD READ (cont.)

Values obtained by scanning the character strings in a specified buffer are converted accord-
ing to the specified format and stored in the proper locations.

9BCDREAD provides 9EDIT with a format, a buffer (and its size), and the location of routine
(part of 9BCDREAD) that will input records for 9EDIT.

6EEFLAG signals whether an END= or ERR= has been specified: if so, and an EOF or error
occurs, 9BCDREEE assumes control.

Size: 48
Subroutines Used: 9EDIT (072), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092),
8TERROR (093), 8TEDIT (094)
(Indirectly): 8TO (051), 7ERROR (066), 9IODATA (074), 9STOP (088),
9BINDEC (089), 7GETMODE (090)

9BCDWRIT (9PRINT), BCD WRITE

This routine sets up the format scan routine (9EDIT) to interface directly with the user pro-
gram in obtaining locations of list items, converting them into character strings according to the
specified format, and placing these strings into a buffer for output.

9BCDWRIT provides POEDIT with a format, a buffer (and its size), and the location of a
routine (part of 9BCDWRIT) that will output records for 9EDIT.

The output part of 9BCDWRIT takes care of vertical format control (based on the character in
column 1) if the L (list) option has been specified on the DCB being used.
9BCDWRITE (PRINT), BCD WRITE (cont.)

Size: 57

Subroutines Used: 9IEDIT (072), 7UNITADR (080), 8ERROR (093), 8TEDIT (094)

(Indirectly): 8TO (051), 7ERROR (066), 9IODATA (074), 7BINDEC (089), 7GETMODE (090), 8TINIT (092)

9BINREAD (9BINWRIT) (7BINREAD), BINARY READ AND WRITE

Calling Sequence: (AI) = unit number
BAL, LL 9BINREAD/9BINWRITE

Purpose: 9BINREAD: BINARY READ subroutine for such statements as READ (unit) list
9BINWRIT: BINARY WRITE subroutine for such statements as WRITE (unit) list
7BINREAD: Special entry point used by 9BINRDEE, which processes END= and ERR=, e.g., WRITE (unit,END=5, ERR=6) list

Each binary (also called intermediate) READ or WRITE statement processes exactly one logical record, which may be subdivided into any number of physical records having the following format:

<table>
<thead>
<tr>
<th>Word</th>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>First control word</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>X'3C' = Not last physical record</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>X'1C' = Last physical record</td>
</tr>
<tr>
<td></td>
<td>2 + 3</td>
<td>Byte checksum</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of data bytes in record + 8 (i.e., includes control words)</td>
</tr>
</tbody>
</table>

If the total record is too small, filler is introduced after the data words and before the final control word. In this case, the byte count (in the first control word) does not reflect the total size of the record; in all other cases, it does.

This routine interfaces directly with the user via 9DATA and 9IODATA. When the user has provided the last input/output datum he enters 9ENDIOL, which in turn calls READFIN or WRITEFIN to end the job.

The checksum used is the byte sum (ignoring byte overflow) of all the bytes in the record, including the control words except for the checksum byte. This includes the random filler that may be introduced if too little data is being written.

6EEFLAG signals whether an END= or ERR= has been specified: if so, and an EOF or error occurs, 9BINRDEE assumes control.

Size: 225

Subroutines Used: 8TO (051), 7ERROR (066), 9IODATA (074), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092), 8ERROR (093), 8TEDIT (094)

(Indirectly): 9STOP (088), 7BINDEC (089), 7GETMODE (090)
Calling Sequence: (AI) = number of characters per internal record
(FP) = starting location of format (word address)
(XS) = starting location of internal buffer (words)
BAL, LL 9DECODE/9ENCODE

Purpose: Implements the DECODE and ENCODE statements, using the format scan routine
(9EDIT + 9OEDIT).

Like 9BCDREAD/9BCDWRIT, 9DECODE/9ENCODE provide 9EDIT/9OEDIT with the addresses
of a FORMAT, a buffer (and its size), and a routine to "transfer" records. In the case of read
and write, this routine actually performs an input/output operation. In the case of decode
and encode, however, all it does is increment the buffer location by the size that the user has
specified, in order to step to the next internal "record".

Size: 35
Subroutines Used: 7ERROR (066), 9EDIT (072), 8TERROR (093), 8TEDIT (094)
(Indirectly): 8T0 (051), 9IODATA (074), 7BINDEC (089), 7GETMODE (090)

Calling Sequence: AI = number of elements
BAL, LL 9IOLUSA
(type, E ARG)

where
type = array type (INTG, SNGL, etc.)
ARG = location of the first element (may specify indirect or indexing)
,E = end of the I/O list (optional)

Purpose: Transmits all elements of an unsubscripted array that appears in an I/O list, and
retains the contents of the alpha set of registers.

Size: 35
Subroutines Used: 9IODATA (074), 8TERROR (093), 8TEDIT (094)
(Indirectly): 8T0 (051), 7ERROR (066), 7GETMODE (090), 8TINIT (092)
I/O LIST INDIVIDUAL ITEM TRANSMITTER

Calling Sequence:
For 9IODATA: BAL, LL 9IODATA
    type1.c ARG1
    type2.c ARG2
    ...
    typen.x ARGn

where
- type = type code (INTG, SNGL, etc.,)
- ARG = datum address (may specify indirect or indexing)
- X = E (optional) (If used, 9IODATA will call 9ENDIOL after transmitting the last datum; otherwise, 9IODATA will return to the program, which then calls 9ENDIOL itself or makes additional calls on 9IODATA (or 9IOLUSA).

For 9DATA: BAL, LL 9DATA
(9DATA may be made to return the previous datum again by calling it with the EXCESS DATA TRIGGER (X'40' bit of register 14) set to 1. The beta register set will be retained (except for the use of a few bits of the trigger register).

For 9ENDIOL: BAL, LL 9ENDIOL

Purpose: 9IODATA: Transmits one or more I/O list items to I/O processing routines and retains the alpha set of registers.
9DATA: Obtains the next datum from the I/O list (complex (single or double precision) data are split up and passed in two parts) and places the datum location (as a word address) in 9IODADDR, the datum type (as a code between 1 and 6) in bits 1 through 31 of 9IODTYPE, and the protection bit in bit 0 of 9IODTYPE.
9IODATA sets '8IOTRIG' true to allow 9IODATA and 9ENDIOL to receive calls.

Purpose: 9ENDIOL: Signals the end of an I/O list and retains the alpha set of registers.

Size: 132
Subroutines Used: 7UNITADR (080) 7ERROR (066), 7BINDEC (089), 8TO (051), 8TINIT (092), 8TERROR (093), 8TEDIT (094)

REWIND, REWIND SEQUENTIAL FILE

Calling Sequence: (AI) = unit number
BAL, LL 9REWIND

Purpose: Rewinds any sequential file.

Size: 14
Subroutines Used: 7UNITADR (080)
(Indirectly): 8TO (051), 7ERROR (066), 7BINDEC (089), 8TO (051), 8TINIT (092), 8TERROR (093), 8TEDIT (094)
9BKSPACE, BACKSPACE ONE LOGICAL RECORD

Calling Sequence: \( (A1) = \text{unit number (integer value)} \)
BAL, LL 9BKSPACE

Purpose: Backspaces one logical record on a sequential file (usually a magnetic tape or a RAD). In BCD (EBCDIC), this is simply one physical unit record; in binary, however, a logical record is everything output by one WRITE statement, and may consist of several physical records having the following format:

<table>
<thead>
<tr>
<th>Word</th>
<th>Byte</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td>First control word</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>X'3C' = not last physical record</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>X'1C' = last physical record</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>Byte checksum</td>
</tr>
<tr>
<td>2 + 3</td>
<td></td>
<td>Number of data bytes in record + 8 (i.e., includes control words)</td>
</tr>
<tr>
<td>1-N</td>
<td></td>
<td>Data words (may be none)</td>
</tr>
<tr>
<td>N+1</td>
<td></td>
<td>Second control word</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>Same as word 0, byte 0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>X'BD' (Special binary code)</td>
</tr>
<tr>
<td>2+3</td>
<td></td>
<td>Physical record number (starts at zero)</td>
</tr>
</tbody>
</table>

Thus, in binary it is necessary to read in reverse, picking up only the first word encountered (the last word on the record); bytes 2-3 of this word indicate how many more physical records to back over.

Sigma tapes have only one mode: essentially there is no BCD mode, so the distinction is made only by the software. If bytes 0-1 of the last control word contain X'3CBD' or X'1CBD', the record is binary; otherwise, it is BCD. Unlike the 9-series, backspace works on single, binary, physical records produced by nonstandard means (e.g., BUFFER OUT) provided they do not contain the four hexadecimal digits above.

Note that it is possible to construct a BCD record that looks like a binary record, but it involves the use of two nongraphic characters in the last word. Backspace does not work on such a record, nor does it have any effect if tape is positioned at the load point.

Size: 38

Subroutines Used: BTO (051), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092), 8TERROR (093), 8TEDIT (094)
(Indirectly): BTO (051), 7BINDEC (089), 8TINIT (092)

9ENDFILE, WRITE END-OF-FILE

Calling Sequence: \( (A1) = \text{unit number} \)
BAL, LL 9ENDFILE

Purpose: Writes end-of-file

Size: 14

Subroutines Used: 7UNITADR (080), 8TERROR (093), 8TEDIT (094)
(Indirectly): BTO (051), 7ERROR (066), 7BINDEC (089), 8TINIT (092)

7EOFABRT, END-OF-FILE ABORT

Calling Sequence: B 7EOFABRT

Purpose: When a read or backspace routine encounters an end-of-file for which no special end-of-file provision (i.e., with EOFSET or END=) has been made, 7EOFABRT prints the error message END-OF-FILE ON UNIT N and branches to STOP, which exits to the Monitor.

Size: 33

Subroutines Used: 7ERROR (066), 9STOP (088), 8TERROR (093), 8TEDIT (094)
(Indirectly): BTO (051), 7BINDEC (089), 8TINIT (092)
Calling Sequence: (AI) = unit number
               (EN) = location of entry name
               BAL, LL  7UNITADR

Purpose: Locates the DCB that corresponds to a particular unit number and provides that
         address to the calling I/O routine. First the unit number is truncated to 16 bits (the max-
        imum permissible number is 9999) and stored in BUNITVAL. This cell is used, for example,
         by EOFSET, which needs access to the value of the unit number currently in use.

Next the value is converted from binary to decimal (using 7BINDEC) and stored in BUNITNAM
as a TEXTC character string 2 words in length. This information is used by EOFSET as well as
by 7UNITADR itself when it has to output an error message.

Then the same character string (without the TEXTC count in front) is stored into DCBNAME
preceded by the two characters "F:". For example, at this point the unit number 108 would
appear in BUNITNAM as 03F1F08 and 40404040, while in DCBNAME it would be 067AF1F0
and 40404040. DCBNAME is now used to search the Monitor DCB table and determine if a
DCB by that name exists. The address of the Monitor DCB table is found in the job-associated
TCB.

If the required DCB is found, its address is stored into DCBADR and control returns to the
calling I/O program. If no DCB is found, the message I/O UNIT n IS UNASSIGNED
is output (via 7ERROR) and the job is aborted.

Size: 64

Subroutines Used: 7ERROR (066), 7BINDEC (089), 8ERROR (093), 8TEDIT (094)
(Indirectly): 8T0 (051), 8TINIT (092)

---

Calling Sequence: (FP) = location of ASSIGNED variable
               BAL, LL  9ASFORM

Purpose: Used with statements such as
         WRITE(108, M) list

where M has been ASSIGNED the statement number of a FORMAT statement.

9ASFORM verifies that the variable pointed to has indeed been ASSIGNED and, if so, puts
the location of the start of the FORMAT into register FP.

If the variable has not been ASSIGNED (as determined by looking for a Branch instruction
in the upper 15 bits), 9ASFORM, instead of producing an error, assumes that the location it
has been given is actually the beginning of a FORMAT stored in an array (or scalar). Thus,
if an array element or scalar contains a FORMAT string (which is not strictly legal), 9ASFORM
will function sensibly. If the variable was neither ASSIGNED nor Intended to be the start of a
FORMAT itself, the error produced is a missing left parenthesis in a FORMAT.

Size: 8

Subroutines Used: (none)
(Indirectly): (none)
9ASGOTO, ASSIGNED GO TO

Calling Sequence: \((X7) = \text{location of ASSIGNED variable}\)

Purpose: An assigned variable should contain a direct, unindexed branch instruction. If it does, 9ASGOTO transfers to the address specified in the branch; otherwise, it gives a diagnostic and aborts.

Size: 29

Subroutines Used: 7ERROR (066), 8ERROR (093)

(Indirectly): 8TINIT (092)

9IFSWITCH, TEST SENSE SWITCH

Calling Sequence: \((AI) = \text{switch number (integer value)}\)
BAL, LL 9IFSWITCH
B switch is on (SET)
B switch is off (RESET)

Purpose: Tests sense switches, which are not hardware but are simulated by the Monitor and can be set and reset by the operator key-in SWITCH. For example,

\(15\text{SWITCH} 0000,\text{(SET, 1, 4), (RESET, 2)}\)

where

0000 = the SYSTEM ID that is typed out on the OC device at the start of the job.

Switches can also be initialized (SET or RESET) by the SWITCH control card, which has the same format as the key-in except that the system ID is not specified. Normally, the switches are all reset initially.

The sense switches are kept in relative location 12 of the TCB, bits 26 through 31.

Size: 24

Subroutines Used: 7ERROR (066)
(Indirectly): 8TINIT (092), 8ERROR (093)

9SNSLITE (9IFSLITE), SENSE LIGHT SET AND TEST

Calling Sequence: For 9SNSLITE:
\((AI) = \text{sense light number}\)
BAL, LL 9SNSLITE

For 9IFSLITE:
\((AI) = \text{sense light number}\)
BAL, LL 9IFSLITE
B sense light on (set)
B sense light off (reset)

Purpose: Sets and tests sense lights, which are maintained in memory in a cell named 8SENLITE. For compatibility purposes there are 24 lights, numbered from right to left starting at the right end of the word.

All sense lights are initialized off in 9INITIAL and, after every test by 9IFSLITE, are turned off; in addition, all lights can be turned off by calling 9SNSLITE with a value of zero. For 9IFSLITE, sense light zero is always off.

Size: 30

Subroutines Used: 7ERROR (066), 8TINIT (092)

(Indirectly): 8ERROR (093)
705085  9IFOVEL, FLOWING OVERFLOW TEST

Calling Sequence: BAL, LL  9IFOVEL
B  set (floating overflow)
B  reset (no floating overflow)

Purpose: Tests for floating overflow. Whenever a real or double precision overflow (including divide by zero) occurs, 8FLOVTRG (the FORTRAN floating overflow trigger) is set in the 9INITIAL trap handler. This is done consistently throughout the system: that is, not only in the generated floating point instructions, but also in the math routines (via 9ERROR) and in BCD input (9EDIT). Complex and double complex operations are also covered, since they are done in real and double precision pieces. After the test, the trigger is turned off again.

Size: 4
Subroutines Used: 8TINIT (092)
(Indirectly): (none)

705086  9UNDEFLB, UNDEFINED LABEL ABORT

Calling Sequence: BAL, LL  9UNDEFLB

Purpose: Prints the error message UNDEFINED LABEL REFERENCED and aborts whenever an undefined label is referenced in a compiled program.

Size: 13
Subroutines Used: 7ERROR (066)
(Indirectly): 8TINIT (092), 8TERROR (093)

705087  9PAUSE, PAUSE

Calling Sequence: (AI) BAL, LL  9PAUSE

Purpose: Types out (on the OC device) *
PAUSE* N

where
N = contents of AI

and then addresses the OC device for input, which is typically just a new line although a maximum of 80 characters may be typed first.

Size: 20
Subroutines Used: 7BINDEC (089), 8TERROR (093)
(Indirectly): 8TO (051)

705088  9STOP, (7STOP), STOP

Calling Sequence: For 9STOP: (AI) BAL, LL  9STOP
For 7STOP: positive decimal integer BAL, LL  9STOP

Purpose: 9STOP: Prints out (on the LO device)

*STOP* N

where
N = contents of AI

and then falls into 7STOP.

7STOP: (Special entrance that types the final message and exits to the Monitor.) Types out (on the OC device) the exit execution time (to the nearest minute) plus the date and then terminates the job through M:EXIT.

Size: 26
Subroutines Used: 7BINDEC (089), 8TERROR (093)
(Indirectly): 8TO (051)
705089  7BINDEC, BINARY TO DECIMAL FOR MESSAGES

Calling Sequence: (A1) = nonnegative integer
(X4) = byte address at which to begin generating string
BAL, LL  7BINDEC

Purpose: Converts the value in A1 to a left-justified string expressed in decimal; since it is
left-justified, it is variable in length (like widthless I format). Zero becomes the one digit 0.

7BINDEC returns with X4 = byte address of byte immediately following the last byte generated.

7BINDEC is used by 9PAUSE, 9STOP, and 7UNITADR.

Size: 13

Subroutines Used: 8TO (051)

(Indirectly): (none)

705090  7GETMODE, ARGUMENT MODE CALCULATION

Calling Sequence: X4 = calling argument word
BAL, LL  7GETMODE

Purpose: Examines the mode and protection bits of a standard calling sequence argument,
and converts it to a word containing the protection bit in bit 0, and an integer code repre­
senting the argument type in bits 1 through 31. The argument is of the form

Bit No: 0 1 2 3 4 5 6 7 8 ...
Use: ------ L K C D R I P

Type code values are

0  no MODE bits set
1  integer
2  real
3  double
4  complex
5  double complex
6  logical

7GETMODE returns with protection bit and type code in M; other registers are not altered.

Size: 12

Subroutines Used: 8TO (051)

(Indirectly): (none)
Additional Names for Library Temps

Calling Sequence: N/A
Purpose: Defines additional names for library temps. Since external references can have addends at no cost, the library should reference the general temp area as 8TO+1, 8TO+2, etc. However, there are still numerous routines in the library that reference them as 8TO, 8T1, 8T2, etc., thus requiring the loading of this program. After the entire library has been converted to reference only 8TO, this program will be removed.

This defines the names 8T1 through 8T17.

Size: 1
Subroutines Used: 8TO (051)
(Indirectly): (none)

Temps for 9INITIAL

Calling Sequence: N/A
Purpose: Provides the temps and special cells that are always loaded because they are used by 9INITIAL. (For other temps see 8TEDIT, 8TERROR, and 8TO.)

Size: 14
Subroutines Used: (none)
(Indirectly): (none)

Temps for I/O

Calling Sequence: N/A
Purpose: Contains most of the temporary storage areas used by the I/O routines in the library. (For other temps see 8TINIT, 8TERROR, and 8TO.)

Size: 151
Subroutines Used: (none)
(Indirectly): (none)
705101  ALOG, DRIVER FOR 9ALOG

Calling Sequence: LI, NA  1
                   BAL, LC  name
                   SNGL  ARG

Purpose: Standard receiving sequence to provide basic external version of real natural logarithm.

Size: 5

Subroutines Used: 9ALOG (001), 8T0 (051), 9SETUP1 (060)
(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705102  DLOG, DRIVER FOR 9DLOG

Calling Sequence: LI, NA  1
                   BAL, LC  name
                   SNGL  ARG

Purpose: Standard receiving sequence to provide basic external version of double precision natural logarithm.

Size: 5

Subroutines Used: 9DLOG (002), 8T0 (051), 9SETUP1 (060)
(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705103  EXP, DRIVER FOR 9EXP

Calling Sequence: LI, NA  1
                   BAL, LC  name
                   SNGL  ARG

Purpose: Standard receiving sequence to provide basic external version of real exponential (e ** arg).

Size: 5

Subroutines Used: 9EXP (003), 8T0 (051), 9SETUP1 (060)
(Indirectly): 9ERROR (065), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705104  DEXP, DRIVER FOR 9DEXP

Calling Sequence: LI, NA  1
                   BAL, LC  name
                   DOUB  ARG

Purpose: Standard receiving sequence to provide basic external version of double precision exponential (e ** arg).

Size: 5

Subroutines Used: 9DEXP (004), 8T0 (051), 9SETUP1 (060)
(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
**705105**  
SIN, DRIVER FOR 9SIN

**Calling Sequence:**  
LI, NA 1  
BAL, LC name  
SNGL ARG

**Purpose:** Standard receiving sequence to provide basic external version of real sine of angle in radians.

**Size:** 5

**Subroutines Used:** 9SIN (005), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

---

**705106**  
DSIN, DRIVER FOR 9DSIN

**Calling Sequence:**  
LI, NA 1  
BAL, LC name  
DOUB ARG

**Purpose:** Standard receiving sequence to provide basic external version of double precision sine of angle in radians.

**Size:** 5

**Subroutines Used:** 9DSIN (006), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

---

**705107**  
ATAN, DRIVER FOR 9ATAN1, 9ATAN2

**Calling Sequence:**  
LI, NA 1  
BAL, LC name or  
SNGL ARG

**Purpose:** Standard receiving sequence to provide basic external version of real arctangent in radians (one or two arguments)

**Size:** 11

**Subroutines Used:** 9ATANI (007), 9SETUPN (062)

(Indirectly): 8TO (051), 9ERROR (066), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

---

**705108**  
DATAN, DRIVER FOR 9DATAN1, 9DATAN2

**Calling Sequence:**  
LI, NA 1  
BAL, LC name or  
DOUB ARG

**Purpose:** Standard receiving sequence to provide basic external version of double precision arctangent in radians (one or two arguments).

**Size:** 11

**Subroutines Used:** 9DATANI (008), 8TO (051), 9SETUPN (062)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
**705109 SQRT, DRIVER FOR 9SQRT**

Calling Sequence:  
\[
\text{LI, NA, 1, BAL, LC, name, SNGL, ARG}
\]

Purpose: Standard receiving sequence to provide basic external version of real square root (positive value).

Size: 5

Subroutines Used: 9SQRT (009), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

**705110 DSQRT, DRIVER FOR 9DSQRT**

Calling Sequence:  
\[
\text{LI, NA, 1, BAL, LC, name, DOUB, ARG}
\]

Purpose: Standard receiving sequence to provide basic external version of double precision square root (positive value).

Size: 5

Subroutines Used: 9DSQRT (010), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

**705111 SINH, DRIVER FOR 9SINH**

Calling Sequence:  
\[
\text{LI, NA, 1, BAL, LC, name, SNGL, ARG}
\]

Purpose: Standard receiving sequence to provide basic external version of real hyperbolic sine.

Size: 5

Subroutines Used: 9SINH (011), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 7ERROR (066), 9ERROR (065), 8T1 (091), 8TINIT (092), 8TERROR (093)

**705112 DSINH, DRIVER FOR 9DSINH**

Calling Sequence:  
\[
\text{LI, NA, 1, BAL, LC, name, DOUB, ARG}
\]

Purpose: Standard receiving sequence to provide basic external version of double precision hyperbolic sine.

Size: 5

Subroutines Used: 9DSINH (012), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9DEXP (004), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705113 TANH, DRIVER FOR 9TANH

Calling Sequence:  
LI, NA  1
BAL, LC  name
SNGL  ARG

Purpose: Standard receiving sequence to provide basic external version of real hyperbolic tangent.

Size: 5

Subroutines Used: 9TANH (013), 9SETUPI (060)

(Indirectly): 9EXP (003), 8T0 (051), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705114 DTANH, DRIVER FOR 9DTANH

Calling Sequence:  
LI, NA  1
BAL, LC  name
DOUB  ARG

Purpose: Standard receiving sequence to provide basic external version of double precision hyperbolic tangent.

Size: 5

Subroutines Used: 9DTANH (014), 8T0 (051), 9SETUPI (060)

(Indirectly): 9DEXP (004), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705115 ASIN, DRIVER FOR 9ASIN

Calling Sequence:  
LI, NA  1
BAL, LC  name
SNGL  ARG

Purpose: Standard receiving sequence to provide basic external version of real arc sine in radians.

Size: 5

Subroutines Used: 9ASIN (015), 8T0 (051), 9SETUPI (060)

(Indirectly): 9ATANI (007), 9SQRT (009), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705116 DASIN, DRIVER FOR 9DASIN

Calling Sequence:  
LI, NA  1
BAL, LC  name
DOUB  ARG

Purpose: Standard receiving sequence to provide basic external version of double precision arc sine in radians.

Size: 5

Subroutines Used: 9DASIN (016), 8T0 (051), 9SETUPI (060)

(Indirectly): 9DATANI (008), 9DSQRT (010), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705117 TAN, DRIVER FOR 9TAN

Calling Sequence: LI, NA 1
                  BAL, LC name
                  SNGL ARG

Purpose: Standard receiving sequence to provide basic external version of real tangent of angle in radians.

Size: 5

Subroutines Used: 9TAN (017), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705118 DTAN, DRIVER FOR 9DTAN

Calling Sequence: LI, NA 1
                  BAL, LC name
                  DOUB ARG

Purpose: Standard receiving sequence to provide basic external version of double precision tangent of angle in radians.

Size: 5

Subroutines Used: 9DTAN (018), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705119 ALOG10, DRIVER FOR 9ALOG10

Calling Sequence: LI, NA 1
                  BAL, LC name
                  SNGL ARG

Purpose: Standard receiving sequence to provide basic external version of real common logarithm (base 10).

Size: 5

Subroutines Used: 9ALOG10 (019), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ALOG (001), 7ERROR (066), 9ERROR (065), 8T1 (091), 8TINIT (092), 8TERROR (093)

705120 DLOG10, DRIVER FOR 9DLOG10

Calling Sequence: LI, NA 1
                  BAL, LC name
                  DOUB ARG

Purpose: Standard receiving sequence to provide basic external version of double precision common logarithm (base 10).

Size: 5

Subroutines Used: 9DLOG10 (020), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DLOG (002), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705121 ACOS, DRIVER FOR 9ACOS

Calling Sequence:
LI, NA
BAL, LC
SNGL, ARG

Purpose: Standard receiving sequence to provide basic external version of real arc cosine in radians.

Size: 5

Subroutines Used: 9ASIN (015), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ATAN1 (007), 9SQRT (009), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TERROR (093), 8TINIT (092)

705122 ATAN2, DRIVER FOR 9ATAN1, 9ATAN2

Calling Sequence:
LI, NA
BAL, LC
SNGL, ARG

Purpose: This is an alternate version of ATAN (705107), provided for compatibility with other FORTRAN systems. Both ATAN and ATAN2 will accept either one or two arguments.

Size: 5

Subroutines Used: 9ATAN1 (007), 8TO (051), 9SETUPN (062)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705123 COS, DRIVER FOR 9COS

Calling Sequence:
LI, NA
BAL, LC
SNGL, ARG

Purpose: Standard receiving sequence to provide basic external version of real cosine of angle in radians.

Size: 5

Subroutines Used: 9SIN (005), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705124 COSH, DRIVER FOR 9COSH

Calling Sequence:
LI, NA
BAL, LC
SNGL, ARG

Purpose: Standard receiving sequence to provide basic external version of real hyperbolic cosine.

Size: 5

Subroutines Used: 9SINH (011), 8TO (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705125  DACOS, DRIVER FOR 9DACOS

Calling Sequence:

LI, NA 1
BAL, LC name
DOUB ARG

Purpose: Standard receiving sequence to provide basic external version of double precision arc cosine in radians.

Size: 5

Subroutines Used: 9DASIN (016), 8T0 (051), 9SETUPI (060)

(Indirectly): 9DATAN1 (008), 9DSQRT (010), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705126  DATAN2, DRIVER FOR 9DATAN1, 9DATAN2

Calling Sequence:

LI, NA 1
BAL, LC name or LI, NA 2
BAL, LC name
DOUB ARG DOUB ARG

Purpose: This is an alternate version of DATAN (705108), provided for compatibility with other FORTRAN systems. Both DATAN and DATAN2 will accept either one or two arguments.

Size: 11

Subroutines Used: 9DATAN1 (008), 8T0 (051), 9SETUPI (062)

(Indirectly): 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705127  DCOS, DRIVER FOR 9DCOS

Calling Sequence:

LI, NA 1
BAL, LC name
DOUB ARG

Purpose: Standard receiving sequence to provide basic external version of double precision cosine of angle in radians.

Size: 5

Subroutines Used: 9DSIN (006), 8T0 (051), 9SETUPI (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705128  DCOSH, DRIVER FOR 9DCOSH

Calling Sequence:

LI, NA 1
BAL, LC name
DOUB ARG

Purpose: Standard receiving sequence to provide basic external version of double precision hyperbolic cosine.

Size: 5

Subroutines Used: 9DSINH (012), 8T0 (051), 9SETUPI (060)

(Indirectly): 9DEXP (004), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
CCOS, DRIVER FOR 9CCOS

Calling Sequence: LI, NA, 1
                  BAL, LC  name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of complex cosine.

Size: 5

Subroutines Used: 9CSIN (040), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 75IN (052), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092),
               8TERROR (093)

CDCOS, DRIVER FOR 9CDCOS

Calling Sequence: LI, NA, 1
                  BAL, LC  name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex cosine.

Size: 7

Subroutines Used: 9CDSIN (041), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9DEXP (004), 9DSIN (006), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092),
               8TERROR (093)

CTANH, DRIVER FOR 9CTANH

Calling Sequence: LI, NA, 1
                  BAL, LC  name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of complex hyperbolic tangent.

Size: 5

Subroutines Used: 9CTAN (042), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 9TANH (013), 9TAN (017), 9ERROR (065), 7ERROR (066),
               8T1 (091), 8TINIT (092), 8TERROR (093)

CDTANH, DRIVER FOR 9CDTANH

Calling Sequence: LI, NA, 1
                  BAL, LC  name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex hyperbolic tangent.

Size: 7

Subroutines Used: 9CDTAN (043), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9DEXP (004), 9DTANH (014), 9DTAN (018), 9ERROR (065), 7ERROR (066),
               8T1 (091), 8TINIT (092), 8TERROR (093)
705133  CACOS, DRIVER FOR 9CACOS

Calling Sequence:  
L, NA 1  
BAL, LC name  
CMPX ARG  

Purpose: Standard receiving sequence to provide basic external version of complex arc cosine.  
Size: 5  
Subroutines Used: 9CASIN (048), 8T0 (051), 9SETUP1 (060)  
(Indirectly): 9DLOG (002), 9DATAN1 (008), 9DSQRT (010), 9CDLOG (037), 9CDSQRT (047), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705134  CDASIN, DRIVER FOR 9CDASIN

Calling Sequence:  
L, NA 1  
BAL, LC name  
KMPX ARG  

Purpose: Standard receiving sequence to provide basic external version of double complex arc sine.  
Size: 7  
Subroutines Used: 9CASIN (048), 8T0 (051), 9SETUP1 (060)  
(Indirectly): 9DLOG (002), 9DATAN1 (008), 9DSQRT (010), 9CDLOG (037), 9CDSQRT (047), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705135  CACOS, DRIVER FOR 9CDACOS

Calling Sequence:  
L, NA 1  
BAL, LC name  
KMPX ARG  

Purpose: Standard receiving sequence to provide basic external version of double complex arc cosine.  
Size: 7  
Subroutines Used: 9CASIN (048), 8T0 (051), 9SETUP1 (060)  
(Indirectly): 9DLOG (002), 9DATAN1 (008), 9DSQRT (010), 9CDLOG (037), 9CDSQRT (047), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705136  CLOG, DRIVER FOR 9CLOG

Calling Sequence:  
L, NA 1  
BAL, LC name  
CMPX ARG  

Purpose: Standard receiving sequence to provide basic external version of complex natural logarithm.  
Size: 5  
Subroutines Used: 9CLOG (036), 8T0 (051), 9SETUP1 (060)  
(Indirectly): 9ALOG (001), 9ATANI (007), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
CDLOG, DRIVER FOR 9CDLOG

Calling Sequence:

<table>
<thead>
<tr>
<th>LL, NA</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC</td>
<td>name</td>
</tr>
<tr>
<td>KMPX</td>
<td>ARG</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of double complex natural logarithm.

Size: 7

Subroutines Used: 9CDLOG (037), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DLOG (002), 9DATAN1 (008), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

CEXP, DRIVER FOR 9CEXP

Calling Sequence:

<table>
<thead>
<tr>
<th>LL, NA</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC</td>
<td>name</td>
</tr>
<tr>
<td>CMPX</td>
<td>ARG</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of complex exponential (e ** arg).

Size: 5

Subroutines Used: 9CEXP (038), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DEXP (004), 9DSIN (006), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

CSIN, DRIVER FOR 9CSIN

Calling Sequence:

<table>
<thead>
<tr>
<th>LL, NA</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC</td>
<td>name</td>
</tr>
<tr>
<td>CMPX</td>
<td>ARG</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of complex sine.

Size: 5

Subroutines Used: 9CSIN (040), 8TO (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 7SIN (052), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705141  CDSIN, DRIVER FOR 9CDSIN

Calling Sequence: L1, NA 1
                  BAL, LC name
                  KMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex sine.

Size: 7

Subroutines Used: 9CDSIN (041), 8TO (051), 9SETUPI (060)
(Indirectly): 9DEXP (004), 9DSIN (006), 9ERROR (065), 7ERROR (066), 8TI (091),
               BTINIT (092), 8TERROR (093)

705142  CTAN, DRIVER FOR 9CTAN

Calling Sequence: L1, NA 1
                  BAL, LC name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of complex tangent.

Size: 5

Subroutines Used: 9CTAN (042), 8TO (051), 9SETUPI (060)
(Indirectly): 9EXP (003), 9DTANH (014), 9TAN (017), 9ERROR (065), 7ERROR (066),
               8TI (091), BTINIT (092), 8TERROR (093)

705143  CDTAN, DRIVER FOR 9CDTAN

Calling Sequence: L1, NA 1
                  BAL, LC name
                  KMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex tangent.

Size: 7

Subroutines Used: 9CDTAN (043), 8TO (051), 9SETUPI (060)
(Indirectly): 9DEXP (004), 9DTANH (014), 9TAN (017), 9ERROR (065), 7ERROR (066),
               8TI (091), BTINIT (092), 8TERROR (093)

705144  CATAN, DRIVER FOR 9CATAN

Calling Sequence: L1, NA 1
                  BAL, LC name
                  CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of complex arc-tangent.

Size: 5

Subroutines Used: 9CATAN (044), 8TO (051), 9SETUPI (060)
(Indirectly): 9ALOG (001), 9ATAN (007), 9ERROR (065), 7ERROR (066), 8TI (091),
               BTINIT (092), 8TERROR (093)
705145 CDATAN, DRIVER FOR 9CDATAN

Calling Sequence:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of double complex arctangent.

Size: 7

Subroutines Used: 9CDATAN (045), BTO (051), 9SETUP1 (060)

(Indirectly): 9DLOG (002), 9DATAN1 (008), 9ERROR (065), 7ERROR (066), BTO (091), 8TINIT (092), 8TERROR (093)

705146 CSQRT, DRIVER FOR 9CSQRT

Calling Sequence:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of complex square root.

Size: 5

Subroutines Used: 9CSQRT (046), BTO (051), 9SETUP1 (060)

(Indirectly): 9SQR (009), 9ERROR (065), 7ERROR (066), BTO (091), 8TINIT (092), 8TERROR (093)

705147 CDSQRT, DRIVER FOR 9CDSQRT

Calling Sequence:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of double complex square root.

Size: 7

Subroutines Used: 9CDSQRT (047), BTO (051), 9SETUP1 (060)

(Indirectly): 9DSQR (010), 9ERROR (065), 7ERROR (066), BTO (091), 8TINIT (092), 8TERROR (093)

705148 CASIN, DRIVER FOR 9CASIN

Calling Sequence:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Purpose: Standard receiving sequence to provide basic external version of complex arc sine.

Size: 5

Subroutines Used: 9CASIN (048), BTO (051), 9SETUP1 (060)

(Indirectly): 9DLOG (002), 9DSQR (010), 9DATAN1 (008), 9CDSQR (047), 9ERROR (065), 7ERROR (066), BTO (091), 8TINIT (092), 8TERROR (093)
CSINH, DRIVER FOR 9CSINH

Calling Sequence:
LI, NA 1
BAL, LC name
CMPX ARG

Purpose: Standard receiving sequence to provide basic external version of complex hyperbolic sine.

Size: 5

Subroutines Used: 9CSIN (040), 8TO (051), 9SETUPI (060)
(Indirectly): 9SIN (052), 9EXP (003), 9ERROR (065), 7ERROR (066), 8T1 (091),
8TINIT (092), 8TERROR (093)

CCOSH, DRIVER FOR 9CCOSH

Calling Sequence:
LI, NA 1
BAL, LC name
CMPX ARG

Purpose: Standard receiving sequence to provide basic external version of complex hyperbolic cosine.

Size: 5

Subroutines Used: 9CSIN (040), 8TO (051), 9SETUPI (060)
(Indirectly): 9EXP (003), 9SIN (052), 9ERROR (065), 7ERROR (066), 8T1 (091),
8TINIT (092), 8TERROR (093)

CDSINH, DRIVER FOR 9CDSINH

Calling Sequence:
LI, NA 1
BAL, LC name
CMPX ARG

Purpose: Standard receiving sequence to provide basic external version of complex hyperbolic sine.

Size: 7

Subroutines Used: 9CDSIN (041), 8TO (051), 9SETUPI (060)
(Indirectly): 9DSIN (006), 9DEXP (004), 9ERROR (065), 7ERROR (066), 8T1 (091),
8TINIT (092), 8TERROR (093)

CDCOSH, DRIVER FOR 9CDCOSH

Calling Sequence:
LI, NA 1
BAL, LC name
CMPX ARG

Purpose: Standard receiving sequence to provide basic external version of complex hyperbolic cosine.

Size: 7

Subroutines Used: 9CDSIN (041), 8TO (051), 9SETUPI (060)
(Indirectly): 9DEXP (004), 9DSIN (006), 9ERROR (065), 7ERROR (066), 8T1 (091),
8TINIT (092), 8TERROR (093)
705153 FLOAT, DRIVER FOR 9ITOR

Calling Sequence: LI, NA, 1
BAL, LC, name
INTG, ARG

Purpose: Standard receiving sequence to provide basic external version of integer to real conversion.

Size: 5

Subroutines Used: 8TO (051), 9ITOD (055), 9SETUP1 (060)

(Indirectly): (none)

705154 DFLOAT, DRIVER FOR 9ITOD

Calling Sequence: LI, NA, 1
BAL, LC, name
INTG, ARG

Purpose: Standard receiving sequence to provide basic external version of integer to double precision conversion.

Size: 5

Subroutines Used: 8TO (051), 9ITOD (055), 9SETUP1 (060)

(Indirectly): (none)

705155 INT, DRIVER FOR 9RTOI

Calling Sequence: LI, NA, 1
BAL, LC, name
SNGL, ARG

Purpose: Standard receiving sequence to provide basic external version of real to integer conversion.

Size: 5

Subroutines Used: 8TO (051), 9DTOI (056), 9SETUP1 (060)

(Indirectly): (none)

705156 IDINT, DRIVER FOR 9DTOI

Calling Sequence: LI, NA, 1
BAL, LC, name
DOUB, ARG

Purpose: Standard receiving sequence to provide basic external version of double precision to integer conversion.

Size: 5

Subroutines Used: 8TO (051), 9DTOI (056), 9SETUP1 (060)

(Indirectly): (none)
705157  SNGL, DRIVER FOR 9DITOR

Calling Sequence: L1, NA  1
                     BAI, LC  name
                     DOUB  ARG

Purpose: Standard receiving sequence to provide basic external version of double precision
to real conversion.

Size:  5

Subroutines Used: 8TO (051), 9DITOR (057), 9SETUP1 (060)

(Indirectly): (none)

705158  CSNGL, DRIVER FOR 9KTOC

Calling Sequence: L1, NA  1
                     BAI, LC  name
                     KMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex to
complex conversion.

Size:  5

Subroutines Used: 8TO (051), 9KTOC (058), 9SETUP1 (060)

(Indirectly): 9DITOR (057), 8T1 (091)

705159  CABS, DRIVER FOR 9CABS

Calling Sequence: L1, NA  1
                     BAI, LC  name
                     CMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of complex absolute
value (real modulus).

Size:  5

Subroutines Used: 9CSQRT (046), 8TO (051), 9SETUP1 (060)

(Indirectly): 9SQRT (009), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092),
BTERRO (093)

705160  CDABS, DRIVER FOR 9CDABS

Calling Sequence: L1, NA  1
                     BAI, LC  name
                     KMPX  ARG

Purpose: Standard receiving sequence to provide basic external version of double complex
absolute value (double precision modulus).

Size:  7

Subroutines Used: 9CDSQRT (047), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DSQRT (010), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092),
BTERRO (093)
705161  ACOS, DRIVER FOR 9ACOS

Calling Sequence:  LI, NA
                  BAL, LC  name
                  SNGL  ARG

Purpose: This is an alternate version of ACOS (705121), provided for compatibility with
FORTRAN II.

Size: 5

Subroutines Used:  9ASIN (015), 8T0 (051), 9SETUPI (060)
(Indirectly):  9SQR (009), 9ATAN1 (007), 9ERROR (065), 7ERROR (066), 8T1 (091)
               8TINIT (092), 8TErrOR (093)

705162  ARCOS, DRIVER FOR 9ACOS

Calling Sequence:  LI, NA
                  BAL, LC  name
                  SNGL  ARG

Purpose: This is an alternate version of ACOS (705121), provided for compatibility with
IBM 360 and FORTRAN IV.

Size: 5

Subroutines Used:  9ASIN (015), 8T0 (051), 9SETUPI (060)
(Indirectly):  9SQR (009), 9ATAN1 (007), 9ERROR (065), 7ERROR (066), 8T1 (091),
               8TINIT (092), 8TErrOR (093)

705163  ASINF, DRIVER FOR 9ASIN

Calling Sequence:  LI, NA
                  BAL, LC  name
                  SNGL  ARG

Purpose: This is an alternate version of ASIN (705115), provided for compatibility with
FORTRAN II.

Size: 5

Subroutines Used:  9ASIN (015), 8T0 (051), 9SETUPI (060)
(Indirectly):  9ATAN1 (007), 9SQR (009), 9ERROR (065), 7ERROR (066), 8T1 (091),
               8TINIT (092), 8TErrOR (093)

705164  ARSIN, DRIVER FOR 9ASIN

Calling Sequence:  LI, NA
                  BAL, LC  name
                  SNGL  ARG

Purpose: This is an alternate version of ASIN (705115), provided for compatibility with
IBM 360 and FORTRAN IV.

Size: 5

Subroutines Used:  9ASIN (015), 8T0 (051), 9SETUPI (060)
(Indirectly):  9ATAN1 (007), 9SQR (009), 9ERROR (065), 7ERROR (066), 8T1 (091),
               8TINIT (092), 8TErrOR (093)
ATANF, DRIVER FOR 9ATAN1, 9ATAN2

Calling Sequence: LL, NA 1  LL, NA 2
      BAL, LC name or BAL, LC name
      SNGL ARG      SNGL ARG

Purpose: This is an alternate version of ATAN (705107), provided for compatibility with FORTRAN II. Both ATAN and ATANF will accept either one or two arguments.

Size: 11

Subroutines Used: 9ATANI (007), 8TO (051), 9SETUPN (062)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

COSF, DRIVER FOR 9COS

Calling Sequence: LL, NA 1
      BAL, LC name
      SNGL ARG

Purpose: This is an alternate version of COS (705123), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9SIN (005), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

COSHF, DRIVER FOR 9COSH

Calling Sequence: LL, NA 1
      BAL, LC name
      SNGL ARG

Purpose: This is an alternate version of COSH (705124), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9SINH (011), 8TO (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 9ERROR (066), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

DARCOS, DRIVER FOR 9DACOS

Calling Sequence: LL, NA 1
      BAL, LC name
      DOUB ARG

Purpose: This is an alternate version of DACOS (705125), provided for compatibility with IBM 360.

Size: 5

Subroutines Used: 9DASIN (016), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DSQRT (010), 9DATAN1 (008), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705169 DARSIN, DRIVER FOR 9DASIN

Calling Sequence: LI, NA 1
                  BAL, LC  name
                  DOUB ARG

Purpose: This is an alternate version of DASIN (705118), provided for compatibility with IBM 360.

Size: 5

Subroutines Used: 9DASIN (016), 8TO (051), 9SETUP1 (060)

(Indirectly): 9DATAN1 (008), 9DSQRT (010), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705170 EXPF, DRIVER FOR 9EXP

Calling Sequence: LI, NA 1
                  BAL, LC  name
                  SNGL ARG

Purpose: This is an alternate version of EXP (705103), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9EXP (003), 8TO (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705171 FLOATF, DRIVER FOR 9ITOR

Calling Sequence: LI, NA 1
                  BAL, LC  name
                  INTG ARG

Purpose: This is an alternate version of FLOAT (705153), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 8TO (051), 9ITOD (055), 9SETUP1 (060)

(Indirectly): (none)

705172 IFIX, DRIVER FOR 9RTOI

Calling Sequence: LI, NA 1
                  BAL, LC  name
                  SNGL ARG

Purpose: This is an alternate version of INT (705155), provided for compatibility with other FORTRAN systems.

Size: 5

Subroutines Used: 8TO (051), 9RTOI (056), 9SETUP1 (060)

(Indirectly): (none)
705173 LOG, DRIVER FOR 9ALOG

Calling Sequence:

<table>
<thead>
<tr>
<th>LI, NA 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC.name</td>
</tr>
<tr>
<td>SNGL, ARG</td>
</tr>
</tbody>
</table>

Purpose: This is an alternate version of ALOG (705101), provided for compatibility with IBM 360.

Size: 5

Subroutines Used: 9ALOG (001), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705174 LOG10, DRIVER FOR 9ALOG10

Calling Sequence:

<table>
<thead>
<tr>
<th>LI, NA 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC.name</td>
</tr>
<tr>
<td>SNGL, ARG</td>
</tr>
</tbody>
</table>

Purpose: This is an alternate version of ALOG10 (705119), provided for compatibility with IBM 360.

Size: 5

Subroutines Used: 9ALOG10 (019), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ALOG (001), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705175 SINF, DRIVER FOR 9SIN

Calling Sequence:

<table>
<thead>
<tr>
<th>LI, NA 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC.name</td>
</tr>
<tr>
<td>SNGL, ARG</td>
</tr>
</tbody>
</table>

Purpose: This is an alternate version of SIN (705105) provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9SIN (005), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705176 SINHF, DRIVER FOR 9SINH

Calling Sequence:

<table>
<thead>
<tr>
<th>LI, NA 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAL, LC.name</td>
</tr>
<tr>
<td>SNGL, ARG</td>
</tr>
</tbody>
</table>

Purpose: This is an alternate version of SINH (705111), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9SINH (011), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)
705177  SQRTF, DRIVER FOR 9SQRT

Calling Sequence: LI, NA 1
                 BAL, LC name
                 SNGL  ARG

Purpose: This is an alternate version of SQRT (705109), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9SQRT (009), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705178  TANF, DRIVER FOR 9TAN

Calling Sequence: LI, NA 1
                 BAL, LC name
                 SNGL  ARG

Purpose: This is an alternate version of TAN (705115), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9TAN (017), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705179  TANHF, DRIVER FOR 9TANH

Calling Sequence: LI, NA 1
                 BAL, LC name
                 SNGL  ARG

Purpose: This is an alternate version of TANH (705113), provided for compatibility with FORTRAN II.

Size: 5

Subroutines Used: 9TANH (013), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9EXP (003), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705180  ABS, REAL ABSOLUTE VALUE

Calling Sequence: LI, NA 1
                 BAL, LC name
                 SNGL  ARG

Purpose: Real absolute value (basic external version).

Size: 4

Subroutines Used: 9SETUP1 (060)

(Indirectly): (none)
AMAG, REAL IMAGINARY PART

Calling Sequence: L, NA 1
                BAL, LC name
                CMPX ARG

Purpose: Imaginary part of complex argument (basic external version).
Size: 4
Subroutines Used: 9SETUP1 (060)
(Indirectly): (none)

AINT, REAL INTEGRAL VALUE

Calling Sequence: L, NA 1
                BAL, LC name
                SNGL ARG

Purpose: Integer part of argument (fractional part truncated) (basic external version).
Size: 6
Subroutines Used: 9SETUP1 (060)
(Indirectly): (none)

AMAX, REAL MAXIMUM VALUE

Calling Sequence: L, NA n
                BAL, LC (name)
                SNGL ARG1
                SNGL ARG2
                ... ...
                SNGL ARGn

Purpose: Greatest argument (basic external version).
Size: 16
Subroutines Used: 8T0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), 8T1 (091)

AMAX1, REAL MAXIMUM VALUE

Calling Sequence: L, NA 1
                BAL, LC (name)
                SNGL ARG1
                SNGL ARG2
                ... ...
                SNGL ARGn

Purpose: This is an alternate version of AMAX (705183), provided for compatibility with other FORTRAN systems.
Size: 16
Subroutines Used: 8T0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), 8T1 (091)
705185 AMAXO, REAL MAXIMUM VALUE OF INTEGERS

Calling Sequence: LI, NA n
BAL, LC (name)
INTG ARG_1
INTG ARG_2

Purpose: Conversion of greatest integer argument to real value (basic external version).
Size: 17
Subroutines Used: BT0 (051), 9ITOD (055), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)

705186 AMIN, REAL MINIMUM VALUE

Calling Sequence: LI, NA n
BAL, LC (name)
SNGL ARG_1
SNGL ARG_2

Purpose: Smallest argument (basic external version).
Size: 16
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)

705187 AMINI, REAL MINIMUM VALUE

Calling Sequence: LI, NA n
BAL, LC (name)
SNGL ARG_1
SNGL ARG_2

Purpose: This is an alternate version of AMIN (705186), provided for compatibility with other FORTRAN systems.
Size: 16
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)

705188 AMINO, REAL MINIMUM VALUE OF INTEGERS

Calling Sequence: LI, NA n
BAL, LC (name)
INTG ARG_1
INTG ARG_2

Purpose: Conversion of smallest integer argument to real value (basic external version).
Size: 17
Subroutines Used: BT0 (051), 9ITOD (055), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)
AMOD, REAL REMAINDER (MODULO)

Calling Sequence: LI, NA
                  BAL, LC
                  SNGL ARG1
                  SNGL ARG2

Purpose: Arg1 (mod arg2): evaluated as arg1 - arg2*AINT(arg1/arg2), i.e., the sign is the same as arg1. Function is undefined if arg2 = 0 (basic external version).

Subroutines Used: 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

CDBLE, COMPLEX TO DOUBLE COMPLEX

Calling Sequence: LI, NA
                  BAL, LC
                  CMPX name

Purpose: Conversion from complex to double complex (basic external version).

Size: 7

Subroutines Used: 9SETUP1 (060)

(Indirectly): (none)

CDINT, DOUBLE COMPLEX INTEGRAL VALUE

Calling Sequence: LI, NA
                  BAL, LC
                  KMPX name

Purpose: Double complex number formed by the integer values of the real and imaginary parts of double complex argument (basic external version).

Size: 12

Subroutines Used: 8TO (051), 9SETUP1 (060)

(Indirectly): (none)

CINT, COMPLEX INTEGRAL VALUE

Calling Sequence: LI, NA
                  BAL, LC
                  CMPX name

Purpose: Complex number formed by the integer values of the real and imaginary parts of a complex argument (basic external version).

Size: 8

Subroutines Used: 8TO (051), 9SETUP1 (060)

(Indirectly): (none)
705193 CMPLX, COMPLEX FROM TWO REALS

Calling Sequence: LI, NA

BAL, LC

name

SNGL

ARG1

SNGL

ARG2

Purpose: Conversion of two real numbers to complex: CMPLX(x,y) = x + iy (basic external version).

Size: 6

Subroutines Used: BT0 (051), 9SETUP2 (061)

(Indirectly): BT1 (091)

705194 CONJG, COMPLEX CONJUGATE

Calling Sequence: LI, NA

BAL, LC

name

CMFX

ARG

Purpose: Complex conjugate: CONJG(x + iy) = x - iy

Size: 5

Subroutines Used: 9SETUP1 (060)

(Indirectly): (none)

705195 DABS, DOUBLE PRECISION ABSOLUTE VALUE

Calling Sequence: LI, NA

BAL, LC

name

DOUB

ARG

Purpose: Double precision absolute value (basic external version).

Size: 4

Subroutines Used: BT0 (051), 9SETUP1 (060)

(Indirectly): (none)

705196 DBLE, REAL TO DOUBLE CONVERSION

Calling Sequence: LI, NA

BAL, LC

name

SNGL

ARG

Purpose: Conversion of real argument to double precision (basic external version).

Size: 5

Subroutines Used: BT0 (051), 9SETUP1 (060)

(Indirectly): (none)
705197 DCMPLX, DOUBLE COMPLEX FROM TWO DOUBLES

Calling Sequence: LI, NA 2
                  BAL, LC  name
                  DOUB  ARG1
                  DOUB  ARG2

Purpose: Conversion of two double precision numbers to double complex:
DCMPLX(x, y) = x + iy (basic external version).

Size: 6

Subroutines Used: 8TO (051), 9SETUP2 (061)
(Indirectly): 8T1 (091)

705198 DCONJG, DOUBLE COMPLEX CONJUGATE

Calling Sequence: LI, NA 1
                  BAL, LC  name
                  COMX  ARG

Purpose: Double complex conjugate: DCONJG(x + iy) = x - iy (basic external version).

Size: 6

Subroutines Used: 8TO (051), 9SETUP1 (060)
(Indirectly): (none)

705199 DDIM, DOUBLE PRECISION POSITIVE DIFFERENCE

Calling Sequence: LI, NA 2
                  BAL, LC  name
                  DOUB  ARG1
                  DOUB  ARG2

Purpose: Double precision positive difference: DDim(x, y) = x - min(x, y) (basic external version).

Size: 9

Subroutines Used: 8TO (051), 9SETUP2 (061)
(Indirectly): 8T1 (091)

705200 DIM, REAL POSITIVE DIFFERENCE

Calling Sequence: LI, NA 2
                  BAL, LC  name
                  SNGL  ARG1
                  SNGL  ARG2

Purpose: Real positive difference: DIM(x, y) = x - min(x, y) (basic external version).

Size: 8

Subroutines Used: 8TO (051), 9SETUP2 (061)
(Indirectly): 8T1 (091)
DIMAG, IMAGINARY PART OF DOUBLE COMPLEX

Calling Sequence: LI, NA  1
                  BAL, LC name
                  KMPX ARG

Purpose: Double precision value of imaginary part of double complex argument (basic external version).

Size: 5

Subroutines Used: 8T0 (051), 9SETUP1 (060)

(Indirectly): (none)

DINT, DOUBLE PRECISION INTEGRAL VALUE

Calling Sequence: LI, NA  1
                  BAL, LC name
                  DOUB ARG

Purpose: Integer part of the argument expressed as a double precision value.

Size: 8

Subroutines Used: 8T0 (051), 9SETUP1 (060)

(Indirectly): (none)

DMAX, DOUBLE PRECISION MAXIMUM VALUE

Calling Sequence: LI, NA  n
                  BAL, LC (name)
                  DOUB ARG1
                  DOUB ARG2
                  ...  ...
                  DOUB ARGn

Purpose: Greatest argument (basic external version).

Size: 19

Subroutines Used: 8T0 (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

DMAX1, DOUBLE PRECISION MAXIMUM VALUE

Calling Sequence: LI, NA  n
                  BAL, LC (name)
                  DOUB ARG1
                  DOUB ARG2
                  ...  ...
                  DOUB ARGn

Purpose: This is an alternate version of DMAX (705203), provided for compatibility with other FORTRAN systems.

Size: 19

Subroutines Used: 8T0 (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)
705205  DMIN, DOUBLE PRECISION MINIMUM VALUE

Calling Sequence:
LI, NA  n
BAL, LC  (name)
DOUB  ARG_1
DOUB  ARG_2
:  :
DOUB  ARG_n

Purpose: Smallest argument (basic external version).
Size: 19
Subroutines Used: 8TO (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), 8T1 (091)

705206  DMIN1, DOUBLE PRECISION MINIMUM VALUE

Calling Sequence:
LI, NA  n
BAL, LC  (name)
DOUB  ARG_1
DOUB  ARG_2
:  :
DOUB  ARG_n

Purpose: This is an alternate version of DMIN (705205), provided for compatibility with other FORTRAN systems.
Size: 19
Subroutines Used: 8TO (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), 8T1 (091)

705207  DMOD, DOUBLE PRECISION REMAINDER (MODULO)

Calling Sequence:
LI, NA  2
BAL, LC  name
DOUB  ARG_1
DOUB  ARG_2

Purpose: Arg_1 (mod arg_2): evaluated as arg_1 - arg_2 * DINT(arg_1 / arg_2), i.e., the sign is the same as arg_1. Function is undefined if arg_2 = 0 (basic external version).
Size: 12
Subroutines Used: 8TO (051), 9SETUP2 (060)
(Indirectly): 8T1 (091)

705208  DREAL, REAL PART OF DOUBLE COMPLEX

Calling Sequence:
LI, NA  1
BAL, LC  name
KMPX  ARG

Purpose: Real part of double complex argument expressed as double precision value.
Size: 4
Subroutines Used: 8TO (051), 9SETUPI (060)
(Indirectly): (none)
DSIGN, DOUBLE PRECISION TRANSFER OF SIGN
Calling Sequence: LI, NA 2
                           BAL, LC  name
                           DOUB  ARG1
                           DOUB  ARG2

Purpose: Magnitude of arg1 with sign of arg2 (zero is considered positive) (basic external version).
Size: 8
Subroutines Used: BT0 (051), 9SETUP2 (061)
(Indirectly): BT1 (091)

IABS, INTEGER ABSOLUTE VALUE
Calling Sequence: LI, NA 1
                           BAL, LC  name
                           INTG  ARG

Purpose: Integer absolute value (basic external version).
Size: 4
Subroutines Used: BT0 (051), 9SETUP1 (060)
(Indirectly): (none)

IAND, INTEGER BOOLEAN PRODUCT
Calling Sequence: LI, NA n
                           BAL, LC  name
                           INTG  ARG1
                           INTG  ARG2
                           ...
                           INTG  ARGn

Purpose: Integer result of Boolean AND (basic external version).
Size: 14
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)

ICOMPL, INTEGER ONE'S COMPLEMENT
Calling Sequence: LI, NA 1
                           BAL, LC  name
                           INTG  ARG

Purpose: This is an alternate version of INOT (705217), provided for compatibility with other other FORTRAN systems.
Size: 5
Subroutines Used: BT0 (051), 9SETUP1 (060)
(Indirectly): (none)
705213  IDIM, INTEGER POSITIVE DIFFERENCE

Calling Sequence: LI, NA
                  BAL, LC  2
                  INTG  ARG1
                  INTG  ARG2

Purpose: Integer positive difference: IDIM(x,y) = x1 - min (x1,y2)

Size: 8

Subroutines Used: 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

705214  IEOR, INTEGER BOOLEAN EXCLUSIVE OR

Calling Sequence: LI, NA
                  BAL, LC  n
                  INTG  ARG1
                  INTG  ARG2
                    ...
                  INTG  ARGn

Purpose: Integer result of Boolean exclusive OR (basic external version).

Size: 14

Subroutines Used: 8TO (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

705215  IEXCLR, INTEGER EXCLUSIVE OR

Calling Sequence: LI, NA
                  BAL, LC  n
                  INTG  ARG1
                  INTG  ARG2
                    ...
                  INTG  ARGn

Purpose: This is an alternate version of IEOR (705214), provided for compatibility with other FORTRAN systems.

Size: 14

Subroutines Used: 8TO (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

705216  IF, DRIVER FOR 9IFR

Calling Sequence: LI, NA
                  LI, NA  2
                  LI, NA  n
                  BAL, LC  name or BAL, LC  name or
                  INTG  ARG1  INTG  ARG1
                  INTG  ARG2  INTG  ARG2
                    ...  ...
                  INTG  ARGn  INTG  ARGn

Purpose: Test for approximately equal or approximately zero (basic external version). If no third argument is given, the value 2^-18 is used; if no second argument is given, the value zero is used.

Size: 17

Subroutines Used: 8TO (051), 9IFR (053), 9SETUPN (062)

(Indirectly): (none)
### I NOT, INTEGER ONE'S COMPLEMENT

**Calling Sequence:**
```
LI, NA  1
BAL, LC name
INTG    ARG
```

**Purpose:** One's complement (basic external version).

**Size:** 5

**Subroutines Used:** 8T0 (051), 9SETUP1 (060)

(Indirectly): (none)

### I OR, INTEGER BOOLEAN SUM

**Calling Sequence:**
```
LI, NA  n
BAL, LC (name)
INTG    ARG1
INTG    ARG2
:       :
INTG    ARGn
```

**Purpose:** Integer result of Boolean OR (basic external version).

**Size:** 14

**Subroutines Used:** 8T0 (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

### I SIGN, INTEGER TRANSFER OF SIGN

**Calling Sequence:**
```
LI, NA  2
BAL, LC name
INTG    ARG1
INTG    ARG2
```

**Purpose:** Magnitude of arg1 with sign of arg2 (zero is considered positive) (basic external version).

**Size:** 8

**Subroutines Used:** 8T0 (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

### LOCF, LOCATION FUNCTION

**Calling Sequence:**
```
LI, NA  1
BAL, LC LOCF
any    ARG
```

**Purpose:** Integer value of word address location of argument. Not valid for statement numbers (basic external version).

**Size:** 5

**Subroutines Used:** 9SETUP1 (060)

(Indirectly): (none)
705221 MAX, INTEGER MAXIMUM VALUE

Calling Sequence: LI, NA
BAL, LC (name)
INTG ARG₁
INTG ARG₂
... ...
INTG ARGₙ

Purpose: Greatest argument (basic external version).
Size: 16
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062)

705222 MAX0, INTEGER MAXIMUM VALUE

Calling Sequence: LI, NA
BAL, LC (name)
INTG ARG₁
INTG ARG₂
... ...
INTG ARGₙ

Purpose: This is an alternate version of MAX (705221), provided for compatibility with other FORTRAN systems.
Size: 16
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062)

705223 MAX1, INTEGER MAXIMUM VALUE OF REALS

Calling Sequence: LI, NA
BAL, LC (name)
SNGL ARG₁
SNGL ARG₂
... ...
SNGL ARGₙ

Purpose: Conversion of greatest real argument (basic external version).
Size: 17
Subroutines Used: BT0 (051), 9DITO1 (056), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)

705224 MIN, INTEGER MINIMUM VALUE

Calling Sequence: LI, NA
BAL, LC (name)
INTG ARG₁
INTG ARG₂
... ...
INTG ARGₙ

Purpose: Smallest argument (basic external version).
Size: 16
Subroutines Used: BT0 (051), 9SETUPV (063)
(Indirectly): 9SETUPN (062), BT1 (091)
### MINI, INTEGER MINIMUM VALUE

**Calling Sequence:**
- L1, NA  
- BAL, LC  (name)  
- INTG  ARG1  
- INTG  ARG2  
- . . .  
- INTG  ARGn

**Purpose:** This is an alternate version of MIN (705224), provided for compatibility with other FORTRAN systems.

**Size:** 16

**Subroutines Used:** 8TO (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

### MINI, INTEGER MINIMUM VALUE OF REALS

**Calling Sequence:**
- L1, NA  n  
- BAL, LC  (name)  
- SNGL  ARG1  
- SNGL  ARG2  
- . . .  
- SNGL  ARGn

**Purpose:** Conversion of smallest real argument (basic external version).

**Size:** 17

**Subroutines Used:** 8TO (051), 9DTOI (056), 9SETUPV (063)

(Indirectly): 9SETUPN (062), 8T1 (091)

### MOD, INTEGER REMAINDER (MODULO)

**Calling Sequence:**
- L1, NA  2  
- BAL, LC  name  
- INTG  ARG1  
- INTG  ARG2

**Purpose:** Arg1 (mod arg2): evaluated as arg1 - arg2 * INT(arg1/arg2), i.e., the sign is the same as arg1. Function is undefined if arg2 = 0 (basic external version).

**Size:** 8

**Subroutines Used:** 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

### REAL, REAL PART OF COMPLEX

**Calling Sequence:**
- L1, NA  1  
- BAL, LC  name  
- CMPX  ARG

**Purpose:** Real part of complex argument (basic external version).

**Size:** 4

**Subroutines Used:** 8TO (051), 9SETUP1 (060)

(Indirectly): (none)
705229 SIGN, REAL TRANSFER OF SIGN

Calling Sequence: L1, NA
                  BAL, LC
                  SNGL ARG₁
                  SNGL ARG₂

Purpose: Magnitude of arg₁ with sign of arg₂ (zero is considered positive) (basic external version).

Size: 8

Subroutines Used: 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

705230 ABSF, REAL ABSOLUTE VALUE

Calling Sequence: L1, NA
                  BAL, LC
                  SNGL ARG

Purpose: This is an alternate version of ABS (705200), provided for compatibility with FORTRAN II.

Size: 4

Subroutines Used: 8TO (051), 9SETUP1 (060)

(Indirectly): (none)

705231 DIMF, REAL POSITIVE DIFFERENCE

Calling Sequence: L1, NA
                  BAL, LC
                  SNGL ARG₁
                  SNGL ARG₂

Purpose: This is an alternate version of DIM (705200), provided for compatibility with FORTRAN II.

Size: 8

Subroutines Used: 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

705232 SIGNF, REAL MAGNITUDE

Calling Sequence: L1, NA
                  BAL, LC
                  SNGL ARG₁
                  SNGL ARG₂

Purpose: This is an alternate version of SIGN (705229), provided for compatibility with FORTRAN II.

Size: 8

Subroutines Used: 8TO (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)
SSWTCH, SENSE SWITCH TEST

Calling Sequence: 

\[
\text{LL,NA} \quad 1 \quad \text{BAL,LC} \quad \text{SSWTCH} \\
\text{INTG} \quad \text{N} \quad \text{INTG} \quad \text{N} \\
\text{or} \quad 2 \quad \text{LL,NA} \quad \text{BAL,LC} \quad \text{SSWTCH} \\
\text{INTG} \quad \text{J} \quad \text{INTG} \quad \text{J}
\]

where

- \( N \) is the number (1 through 4) of the sense switch to be tested.
- \( J \) is an integer variable into which will be stored the value 1 if the switch is on, or the value 2 if the switch is off.

Purpose: Tests the Monitor-simulated sense switches. SSWITCH may be referenced as a function, as well as called as a subroutine. Used in this way, it returns in AL the value .TRUE. if the switch is on, or the value .FALSE. if the switch is off. This is really a kind of standard driver, because 9IFSWITCH does the real work.

For further information on the use of sense switches, see 9IFSWITCH (083).

Size: 17

Subroutines Used: 8TO (051), 9SETUPN (062), 9IFSWITCH (083)

(Indirectly): 7ERROR (066)

SLITET, SENSE LIGHT TEST

Calling Sequence: 

\[
\text{LL,NA} \quad 1 \quad \text{BAL,LC} \quad \text{SLITET} \\
\text{INTG} \quad \text{N} \quad \text{INTG} \quad \text{N} \\
\text{or} \quad 2 \quad \text{LL,NA} \quad \text{BAL,LC} \quad \text{SLITET} \\
\text{INTG} \quad \text{J} \quad \text{INTG} \quad \text{J}
\]

where

- \( N \) is the number (1 through 24) of the sense light to be tested.
- \( J \) is an integer variable into which will be stored the value 1 if the light is on, or the value 2 if the light is off.

Purpose: Tests the simulated sense lights. SLITET may be used as a function as well as called as a subroutine. Used in this way, it returns in AL the value .TRUE. if the light is on, or the value .FALSE. if the light is off. The sense light is turned off after the test. This is really a kind of standard driver, because 9IFSLITE does the real work.

For further information on the use of sense lights see 9SNSLITE (084).

Size: 17

Subroutines Used: 8TO (051), 9SETUPN (062), 9SNSLITE (084)

(Indirectly): 7ERROR (066), 8TINIT (092)
SLITE, SET SENSE LIGHT

Calling Sequence: LL, NA 1
BAL, LC SLITE
INTG N

where N is the number (0 through 24) of the sense light to be set.
If N is zero, all the sense lights are turned off.

Purpose: Sets the simulated sense lights. SLITE is really a standard driver, because 9SNSLITE does all the work.
For further information on the use of sense lights see 9SNSLITE (084).

Size: 5
Subroutines Used: 9SETUP1 (060), 9SNSLITE (084)
(Indirectly): 7ERROR (066), 8TINIT (092)

OVERFL, TEST FOR FLOATING OVERFLOW

Calling Sequence: LL, NA 1
BAL, LC OVERFL
INTG J

where J is an integer variable into which will be stored the value 1 if overflow has occurred, or the value 2 if it has not.

Purpose: Tests 8FLOVTRG (the FORTRAN floating overflow trigger). After the test, the trigger is turned off. OVERFL may be referenced as a logical function as well as called as a subroutine. Used in this way, it returns in AL the value .TRUE. if overflow has occurred, or the value .FALSE. if it has not.
For further information on floating overflow see 9IFOVFL (085).

Size: 8
Subroutines Used: 9SETUP1 (060), 9IFOVFL (085)
(Indirectly): 8TINIT (092)

EXIT, EXIT TO THE MONITOR

Calling Sequence: LL, NA 0
BAL, LC EXIT

Purpose: Prints *EXIT* on unit 108, then branches to 7STOP, which closes DCBs and exits to the Monitor.

Size: 9
Subroutines Used: 9SETUP0 (059), 9STOP (088)
(Indirectly): 7BINDEC (089), 8TERROR (093)
**705239**  EOFSET, SET END-OF-FILE EXIT

Calling Sequence:  

<table>
<thead>
<tr>
<th>L</th>
<th>LNA</th>
<th>2</th>
<th>BAL, LC</th>
<th>EOFSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTG</td>
<td>LOC</td>
<td>INTG</td>
<td>UNIT</td>
<td></td>
</tr>
</tbody>
</table>

where  

- **LOC** is a statement number or assigned variable to which a transfer will be made on end-of-file.  
- **UNIT** is an integer variable into which will be stored the logical unit number on which the EOF occurred.

**Purpose:** Sets location to transfer to when EOF occurs. Both arguments are optional. If the routine is called with only the LOC argument, the transfer will be made when EOF occurs, but no unit number will be stored. If called with no arguments, resets system to terminate on EOF.

**Size:** 43

**Subroutines Used:** 9SETUPM (062), 7ERROR (066), 7EOFABRT (078), 8TINIT (092), 8TEDIT (094)

(Indirectly): 7ERROR (066), 9STOP (088), 8TERROR (093)

---

**705240**  SETEOF, SET END-OF-FILE EXIT

Calling Sequence:  

<table>
<thead>
<tr>
<th>L</th>
<th>LNA</th>
<th>2</th>
<th>BAL, LC</th>
<th>SETEOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTG</td>
<td>UNIT</td>
<td>INTG</td>
<td>LOC</td>
<td></td>
</tr>
</tbody>
</table>

where LOC and UNIT are the same as described under EOFSET (except note that their order is reversed).

**Purpose:** Performs the same operation as EOFSET. This routine is provided only for compatibility with the 9300 FORTRAN system. It is preferable to use EOFSET.

**Size:** 41

**Subroutines Used:** 9SETUPM (062), 7ERROR (066), 7EOFABRT (078), 8TINIT (092), 8TEDIT (094)

(Indirectly): 7ERROR (066), 9STOP (088), 8TERROR (093)
**BUFFERIN, DIRECT INPUT**

Calling Sequence:

```
L1, NA 6
BAL, LC BUFFERIN
INTG UNIT
INTG MODE
EVRY START
INTG WORDS
INTG INDICATOR
INTG COUNT
```

where

- **UNIT**  is the value of the unit number on which the operation is to be performed.
- **MODE**  is an integer that determines the mode of the operation: if \( \text{MODE} = 0 \), the mode is BCD (EBCDIC); if \( \text{MODE} \neq 0 \), the mode is binary (0 or 1 is customary).
- **START**  is the starting location of the internal buffer.
- **WORDS**  is an integer specifying the number of words to be transferred.
- **INDICATOR**  is an integer variable into which will be stored an indication of the status of the operation:
  - 2 = Normal completion
  - 3 = End-of-file
  - 4 = Error

Since the Batch Processing Monitor (BPM) does not provide end-action, this routine does not function asynchronously. Thus, the indicator never assumes the value 1 (Incomplete).

- **COUNT**  is an optional integer variable into which will be stored the number of words actually transferred.

**Purpose:**

`BUFFERIN` gives the user more direct control over input operations than is possible with formatted or unformatted READ statements, enabling him to process records of any size and format. For more information on the operation and use of `BUFFERIN`, see the Sigma 5/7 FORTRAN IV Reference Manual, SDS 90.09.56.

**Size:** 57

**Subroutines Used:**

- BTO (051), 9SETUPN (062), 7UNITADR (080), 8TEDIT (094)
- (Indirectly): 7ERROR (066), 7BINDEC (089), 8TERROR (093)

---

**BUFFEROU, DIRECT OUTPUT**

Calling Sequence:

```
L1, NA 6
BAL, LC BUFFEROU
INTG UNIT
INTG MODE
EVRY START
INTG WORDS
INTG INDICATOR
INTG COUNT
```

where **UNIT**, **MODE**, **START**, **WORDS**, **INDICATOR**, and **COUNT** are the same as described under `BUFFERIN` (241).

**Purpose:**

`BUFFEROU` gives the user more direct control over output operations than is possible with formatted or unformatted WRITE statements, enabling him to process records of any size and format. For more information on the operation and use of `BUFFEROU`, see the Sigma 5/7 FORTRAN IV Reference Manual, SDS 90.09.56.

Since the compiler truncates names to eight characters, FORTRAN source programs can refer to "BUFFER OUT". The official name however, and the one that must be used by assembly language programs, is "BUFFEROU".

**Size:** 48

**Subroutines Used:**

- BTO (051), 9SETUPN (062), 7UNITADR (080), 8TEDIT (094)
- (Indirectly): 7ERROR (066), 7BINDEC (089), 8TERROR (093)
Calling Sequence: \texttt{LI,NA 2}
\texttt{BAL,LC ABORTSET}
\texttt{INTG LOC}
\texttt{INTG LEVEL}

where

- \texttt{LOC} is a statement number (or assigned variable) to which \texttt{7ERROR} will transfer when an abort level error occurs. A value of zero (instead of a statement number) resets \texttt{7ERROR} to abort to the Monitor.
- \texttt{LEVEL} is an optional integer value from 1 through 15. \texttt{7ERROR} will abort (or transfer to LOC) on any run-time error whose error severity level is greater than or equal to this value.

If \texttt{ABORTSET} is called with only the LOC argument, the abort exit (\texttt{8ABORTEX}) is set up, but the abort severity (\texttt{8ABRTSEV}) is left alone.

Purpose: \texttt{ABORTSET} can be used to do two things:

1. Change the error severity level at which library routines will abort. Certain errors (level 15) always abort. With the standard severity level (8), all of the others recover and continue, unless prohibited by this routine.
2. Obtain control when a serious error occurs, instead of aborting the job. This allows the user to attempt some kind of recovery. No provision is made for finding out what the error was; the user has to anticipate this.

Size: 21

Subroutines Used: \texttt{8TO (051), 9SETUPN (062), 8TINIT (092)}

(Indirectly): (none)