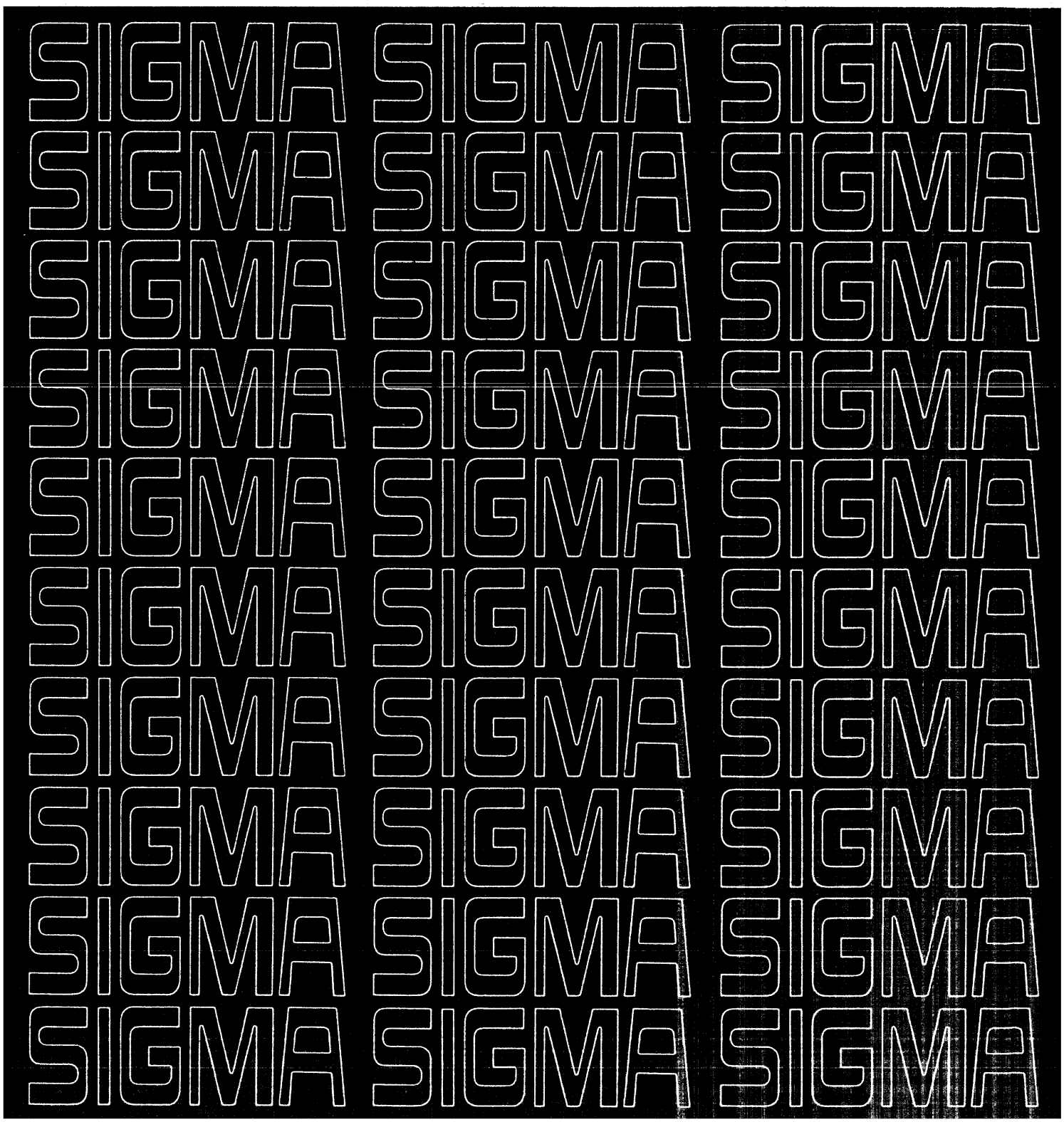
SDS SIGMA 5/7 FORTRAN IV LIBRARY

Technical Manual

SCIENTIFIC DATA SYSTEMS



Price: $3.00

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

Printed in U.S.A.

RELATED PUBLICATIONS

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

CONTENTS

TABLES

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, a REWIND 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 SSWTCH 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 so doing, 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 any CALs it uses near the end of its listing.
2. Automatic loader-generation of DCBs that are REFed.
3. The standard assignments of F:101 through F:106 and F:108.
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. The hardware will correctly set the result to zero and reflect this in the condition codes.

   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

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

REENTRANCY

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 2. Library Routines

Cat. No.	Name of Routine	Other Entries	Size	Description of Routine
705001	9ALOG	7ALOG 7ALOG1 7ALOG2	79	Real natural logarithm
705002	9DLOG	7DLOG 7DLOG1 7DLOG2	116	Double precision natural logarithm
705003	9EXP	7EXP1 7EXP2	59	Real exponential
705004	9DEXP	7DEXP1 7DEXP2	87	Double precision exponential
705005	9SIN	9COS	90	Real sine and cosine
705006	9DSIN	9DCOS	133	Double precision sine and cosine
705007	9ATAN1	9ATAN2	91	Real arctangent
705008	9DATAN1	9DATAN2	121	Double precision arctangent
705009	9SQRT		38	Real square root
705010	9DSQRT		63	Double precision square root
705011	9SINH	9COSH	78	Real hyperbolic sine and cosine
705012	9DSINH	9DCOSH	81	Double precision hyperbolic sine and cosine
705013	9TANH		60	Real hyperbolic tangent
705014	9DTANH		68	Double precision hyperbolic tangent
705015	9ASIN	9ACOS	26	Real arcsine and arccosine
705016	9DASIN	9DACOS	65	Double precision arcsine and arccosine
705017	9TAN		70	Real tangent
705018	9DTAN		110	Double precision tangent
705019	9ALOG10		19	Real common logarithm
705020	9DLOG10		21	Double precision common logarithm
705021	9PWRII		37	Integer raised to an integer power
705022	9PWRRI		35	Real raised to an integer power
705023	9PWRDI		35	Double precision raised to an integer power
705024	9PWRCI		67	Complex raised to an integer power
705025	9PWRKI		74	Double complex raised to an integer power
705026	9PWRRR		57	Real raised to a real power
705027	9PWRDD		57	Double precision raised to a double precision power
705028	9PWRCC		40	Complex raised to a complex power
705029	9PWRKK		41	Double complex raised to a double complex power
705030	9CADD	9CSUB	6	Complex add and subtract
705031	9CMUL		10	Complex multiply
705032	9CDIV		19	Complex divide
705033	9KADD	9KSUB	6	Double complex add and subtract
705034	9KMUL		10	Double complex multiply
705035	9KDIV		17	Double complex divide
705036	9CLOG	7CLOG	56	Complex natural logarithm
705037	9CDLOG	7CDLOG	53	Double complex natural logarithm
705038	9CEXP	7CEXP	20	Complex exponential
705039	9CDEXP	7CDEXP	28	Double complex exponential
705040	9CSIN	9CCOS 9CSINH 9CCOSH	86	Complex sine, cosine, hyperbolic sine, and hyperbolic cosine
705041	9CDSIN	9CDCOS 9CDSINH 9CDCOSH	112	Double complex sine, cosine, hyperbolic sine, and hyperbolic cosine
705042	9CTAN	9CTANH	52	Complex tangent and hyperbolic tangent
705043	9CDTAN	9CDTANH	59	Double complex tangent and hyperbolic tangent
705044	9CATAN		76	Complex arctangent
705045	9CDATAN		104	Double complex arctangent

Table 2. Library Routines (cont.)

Cat. No.	Name of Routine	Other Entries	Size	Description of Routine
705046	9CSQRT	9CABS	88	Complex square root and absolute value (real modulus)
705047	9CDSQRT	9CDABS	92	Double complex square root and absolute value (double precision modulus)
705048	9CASIN	9CACOS 9CDASIN 9CDACOS	91	Complex arcsine and arccosine; double complex arcsine and arccosine
705051	8T0		17	FORTRAN library temp area
705052	7SIN		85	Special SIN/COS/EXP calculations used by various complex functions
705053	9IFR		21	Real approximate equality test
705054	9IFD			Double precision approximate equality test
705055	9ITOD	9ITOR	7	Integer to double precision and integer to real conversions
705056	9DTOI	9RTOI	8	Double precision to integer and real to integer conversions
705057	9DTOR		24	Double precision to real conversion
705058	9KTOC		8	Double complex to complex conversion
705059	9SETUP0		2	Set up zero arguments
705060	9SETUP1		7	Set up one argument
705061	9SETUP2		13	Set up two arguments
705062	9SETUPN	9SETUPM 7SET	14	Set up a fixed number of arguments or a variable number with maximum
705063	9SETUPV		11	Set up a variable number of arguments
705064	9INITIAL		47	Run-time initialization
705065	9ERROR		110	Math library error reporting
705066	7ERROR	7BUFOUT 7BUFOUTC 7ERRHEAD 7ERRINIT 7ERRMARK 7ERRTEXT 7PAC 7PHC 7PRC 7PRL 7PRQ	144	Run-time error reporting
705067	9BCDREAD	9READ 7BCDREAD	48	BCD read
705068	9BCDWRIT	9PRINT	57	BCD write
705069	9BINREAD	9BINWRIT 7BINREAD	225	Binary read and write
705070	9DECODE	9ENCODE	35	Memory-to-memory data conversion
705071	9INPUT			Self-identified (NAME LIST) input
705072	9IEDIT	9OEDIT	1407	FORMAT scan
705073	9IOLUSA		35	Transmit unsubscripted arrays in an I/O list
705074	9IODATA	9DATA 9ENDIOL 7IODATUM	132	Transmit individual items in an I/O list
705075	9REWIND		14	Rewind sequential files
705076	9BKSPACE		38	Backspace one logical record
705077	9ENDFILE		14	Write end-of-file
705078	7EOFABRT		33	End-of-file abort
705079	9RDDISC	9WRDISC		Read and write disc (random access)
705080	7UNITADR		64	Find I/O unit DCB address
705081	9ASFORM		8	Assigned FORMAT
705082	9ASGOTO		29	Assigned GO TO
705083	9IFSWICH		24	Test sense switch
705084	9SNSLITE	9IFSLITE	30	Set and test sense lights

Table 2. Library Routines (cont.)

Cat. No.	Name of Routine	Other Entries	Size	Description of Routine
705085	9IFOVFL		4	Test for floating overflow
705086	9UNDEFLB		13	Undefined label abort
705087	9PAUSE		20	PAUSE
705088	9STOP	7STOP	26	STOP (terminate execution)
705089	7BINDEC		13	Binary to decimal for messages
705090	7GETMODE		12	Argument mode calculation
705091	8T1	8T2	1	Additional names for library temps
		8T3		
		8T4		
		8T5		
		8T6		
		8T7		
		8T8		
		8T9		
		8T10		
		8T11		
		8T12		
		8T13		
		8T14		
		8T15		
		8T16		
		8T17		
705092	8TINIT	8ABORTEX	14	Temps for 9INITIAL
		8ABRTSEV		
		8EOFEXIT		
		8ERREXIT		
		8FLOVTRG		
		8IOTRIG		
		8SENLITE		
705093	8TERROR	8IODLINK	52	Temps for 7ERROR
		8MSGBUF		
		8TALPHA		
		8TBETA		
705094	8TEDIT	6BINBUF	151	Temps for I/O
		6DATLINK		
		6DCBNAME		
		6EEFLAG		
		6EOFCALL		
		6EOFJADR		
		6EOFUADR		
		6EOFUTRG		
		6NRELMTS		
		6RECSIZE		
		8BCDBUF		
		8BUFORG		
		8DCBADR		
		8ENDIOL		
		8INPTERM		
		8IODADDR		
		8IODTYPE		
		8IOENLOC		
		8UNITNAM		
		8UNITVAL		
705101	ALOG		5	Driver for 9ALOG
705102	DLOG		5	Driver for 9DLOG
705103	EXP		5	Driver for 9EXP
705104	DEXP		5	Driver for 9DEXP
705105	SIN		5	Driver for 9SIN

Table 2. Library Routines (cont.)

Cat. No.	Name of Routine	Other Entries	Size	Description of Routine
705106	DSIN		5	Driver for 9DSIN
705107	ATAN		11	Driver for 9ATAN1 and 9ATAN2
705108	DATAN		11	Driver for 9DATAN1 and 9DATAN2
705109	SQRT		5	Driver for 9SQRT
705110	DSQRT		5	Driver for 9DSQRT
705111	SINH		5	Driver for 9SINH
705112	DSINH		5	Driver for 9DSINH
705113	TANH		5	Driver for 9TANH
705114	DTANH		5	Driver for 9DTANH
705115	ASIN		5	Driver for 9ASIN
705116	DASIN		5	Driver for 9DASIN
705117	TAN		5	Driver for 9TAN
705118	DTAN		5	Driver for 9DTAN
705119	ALOG10		5	Driver for 9ALOG10
705120	DLOG10		5	Driver for 9DLOG10
705121	ACOS		5	Driver for 9ACOS
705122	ATAN2		11	Driver for 9ATAN1 and 9ATAN2
705123	COS		5	Driver for 9COS
705124	COSH		5	Driver for 9COSH
705124	DACOS		5	Driver for 9DACOS
705126	DATAN2		11	Driver for 9DATAN1 and 9DATAN2
705127	DCOS		5	Driver for 9DCOS
705128	DCOSH		5	Driver for 9DCOSH
705129	CCOS		5	Driver for 9CCOS
705130	CDCOS		7	Driver for 9CDCOS
705131	CTANH		5	Driver for 9CTANH
705132	CDTANH		7	Driver for 9CDTANH
705133	CACOS		5	Driver for 9CACOS
705134	CDASIN		7	Driver for 9CDASIN
705135	CDACOS		7	Driver for 9CDACOS
705136	CLOG		5	Driver for 9CLOG
705137	CDLOG		7	Driver for 9CDLOG
705138	CEXP		5	Driver for 9CEXP
705139	CDEXP		7	Driver for 9CDEXP
705140	CSIN		5	Driver for 9CSIN
705141	CDSIN		7	Driver for 9CDSIN
705142	CTAN		5	Driver for 9CTAN
705143	CDTAN		7	Driver for 9CDTAN
705144	CATAN		5	Driver for 9CATAN
705145	CDATAN		7	Driver for 9CDATAN
705146	CSQRT		5	Driver for 9CSQRT
705147	CDSQRT		7	Driver for 9CDSQRT
705148	CASIN		5	Driver for 9CASIN
705149	CSINH		5	Driver for 9CSINH
705150	CCOSH		5	Driver for 9CCOSH
705151	CDSINH		7	Driver for 9CDSINH
705152	CDCOSH		7	Driver for 9CDCOSH
705153	FLOAT		5	Driver for 9ITOR
705154	DFLOAT		5	Driver for 9ITOD
705155	INT		5	Driver for 9RTOI
705156	IDINT		5	Driver for 9DTOI
705157	SNGL		5	Driver for 9DTOR
705158	CSNGL		5	Driver for 9KTOC
705159	CABS		5	Driver for 9CABS
705160	CDABS		7	Driver for 9CDABS
705161	ACOSF		5	Driver for 9ACOS
705162	ARCOS		5	Driver for 9ACOS
705163	ASINF		5	Driver for 9ASIN

Table 2. Library Routines (cont.)

Cat. No.	Name of Routine	Other Entires	Size	Description of Routine
705164	ARSIN		5	Driver for 9ASIN
705165	ATANF		11	Driver for 9ATAN1 and 9ATAN2
705166	COSF		5	Driver for 9COS
705167	COSHF		5	Driver for 9COSH
705168	DARCOS		5	Driver for 9DACOS
705169	DARSIN		5	Driver for 9DASIN
705170	EXPF		5	Driver for 9EXP
705171	FLOATF		5	Driver for 9ITOR
705172	IFIX		5	Driver for 9RTOI
705173	LOG		5	Driver for 9ALOG
705174	LOG10		5	Driver for 9ALOG10
705175	SINF		5	Driver for 9SIN
705176	SINHF		5	Driver for 9SINH
705177	SQRTF		5	Driver for 9SQRT
705178	TANF		5	Driver for 9TAN
705179	TANHF		5	Driver for 9TANH
705180	ABS		4	Real absolute value
705181	AIMAG		4	Real imaginary part of complex
705182	AINT		6	Real integral value
705183	AMAX		16	Real maximum value
705184	AMAX1		16	Real maximum value
705185	AMAX0		17	Real maximum value of integer arguments
705186	AMIN		16	Real minimum value
705187	AMIN1		16	Real minimum value
705188	AMIN0		17	Real minimum value of integer arguments
705189	AMOD		11	Real remainder (modulo)
705190	CDBLE		7	Complex to double complex conversion
705191	CDINT		12	Double complex integral value
705192	CINT		8	Complex integral value
705193	CMPLX		6	Complex from two real values
705194	CONJG		5	Complex conjugate
709195	DABS		4	Double precision absolute value
705196	DBLE		5	Real to double precision conversion
705197	DCMPLX		6	Double complex from two double precision values
705198	DCONJG		6	Double complex conjugate
705199	DDIM		9	Double precision positive difference
705200	DIM		8	Real positive difference
705201	DIMAG		5	Double precision imaginary part of double complex
705202	DINT		8	Double precision integral value
705203	DMAX		19	Double precision maximum value
705204	DMAX1		19	Double precision maximum value
705205	DMIN		19	Double precision minimum value
705206	DMIN1		19	Double precision minimum value
705207	DMOD		12	Double precision remainder (modulo)
705208	DREAL		4	Double precision real part of double complex
705209	DSIGN		8	Double precision first argument with sign of second argument
705210	IABS		4	Integer absolute value
705211	IAND		14	Integer Boolean product (AND)
705212	ICOMPL		5	Integer 1's complement (NOT)
705213	IDIM		8	Integer positive difference
705214	IEOR		14	Integer Boolean exclusive OR
705215	IEXCLR		14	Integer Boolean exclusive OR
705216	IF		17	Approximately equal or approximately zero
705217	INOT		5	Integer 1's complement (NOT)

Table 2. Library Routines (cont.)

Cat. No.	Name of Routine	Other Entries	Size	Description of Routine
705218	IOR		14	Integer Boolean sum (OR)
705219	ISIGN		8	Integer first argument with sign of second argument
705220	LOCF		5	Word address of argument
705221	MAX		16	Integer maximum value
705222	MAX0		16	Integer maximum value
705223	MAX1		17	Integer maximum value of real arguments
705224	MIN		16	Integer minimum value
705225	MIN0		16	Integer minimum value
705226	MIN1		17	Integer minimum value of real arguments
705227	MOD		8	Integer remainder (modulo)
705228	REAL		4	Real part of complex
705229	SIGN		8	Real first argument with sign of second argument
705230	ABSF		4	Absolute value
705231	DIMF		8	Positive difference
705232	SIGNF		8	Real first argument with sign of second argument
705233	SSWTCH		17	Test sense switch
705234	SLITET		17	Test sense light
705235	SLITE		5	Set sense light
705236	OVERFL		8	Test for floating overflow
705237	DVCHK		8	Test for floating overflow
705238	EXIT		9	Exit to the Monitor
705239	EOFSET		43	Set up end-of-file exit
705240	SETEOF		41	Set up end-of-file exit
705241	BUFFERIN		52	Direct input
705242	BUFFEROU		43	Direct output
705243	ABORTSET		21	Set up abort exit and severity level

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 $\ell n(X)$,

where

X = FORTRAN IV REAL entity.

Size: 79

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

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (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 0.

Purpose: Calculates $\ell n(X)$,

where

X = FORTRAN IV DOUBLE PRECISION entity.

Size: 116

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

(Indirectly): 7ERROR (066), 8TINIT (092), 8TERROR (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 = FORTRAN IV REAL entity.

Size: 59

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

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

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

705007 9ATAN1 (9ATAN2), REAL ARCTANGENT

Calling Sequence: Uses FORTRAN IV nonstandard calling sequences: 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: 8T0 (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 8-9, X in register pair 10-11 (if 9DATAN2), register 6 is link, result returned in register pair 8-9. 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 and Y = 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 8. Also uses (and does not preserve) registers 2, 9, and 10 (reentrant version also uses register 11).

Purpose: Calculates $\sqrt{X}$,

where

X = 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 8-9, register 6 is link, result returned in register pair 8-9. 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 = 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 8. Also uses (and does not preserve) registers 2, 9, 10, and 11.

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

where

X = 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), 8T1 (091)

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (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, 10, and 11.

Purpose: Calculates tanh(X),

where

X = FORTRAN IV REAL entity.

Size: 60

Subroutines Used: 9EXP (003), 8T0 (051), 8T1 (091)

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

705014 9DTANH, DOUBLE PRECISION HYPERBOLIC 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 tanh(X),

where

X = FORTRAN IV DOUBLE PRECISION entity.

Size: 68

Subroutines Used: 9DEXP (004), 8T1 (091)

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

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

Calling Sequence: Uses FORTRAN IV nonstandard calling sequences: 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), 8T1 (091)

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705016 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 = FORTRAN IV DOUBLE PRECISION entity.

Size: 65

Subroutines Used: 9DATAN1 (008), 9DSQRT (010), 9ERROR (065), 8T1 (091)

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705017 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 = FORTRAN IV REAL entity.

Size: 70

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

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

705018 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 = FORTRAN IV DOUBLE PRECISION entity.

Size: 110

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

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

705019 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 = FORTRAN IV REAL entity.

Size: 19

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

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (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 = FORTRAN IV DOUBLE PRECISION entity.

Size: 21

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

(Indirectly): 8T0 (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 and J = FORTRAN IV INTEGER entities.

Size: 37

Subroutines Used: 9ERROR (065)

(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, register 6 is link, result returned in register 8. Also uses (and does not preserve) registers 2, 10, and 11.

Purpose: Calculate X**I,

where

X = FORTRAN IV REAL entity

I = FORTRAN IV INTEGER entity

Size: 35

Subroutines Used: 9ERROR (065)

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

705023 9PWRDI, 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, 12, and 13.

Purpose: Calculates X**I,

where

X = FORTRAN IV DOUBLE PRECISION entity

I = FORTRAN IV INTEGER entity

Size: 35

Subroutines Used: 9ERROR (065)

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

705024 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

Z = FORTRAN IV COMPLEX entity

I = FORTRAN IV INTEGER entity

Size: 67

Subroutines Used: 9PWRRI (022), 8T0 (051), 9ERROR (065), 8T1 (091)

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

705025 9PWRKI, DOUBLE COMPLEX RAISED TO INTEGER POWER

Calling Sequence: Uses FORTRAN IV nonstandard calling sequence: Z in register pairs 8-9 and 10-11, 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

Z = FORTRAN IV DOUBLE COMPLEX entity

I = FORTRAN IV INTEGER entity

Size: 74

Subroutines Used: 9PWRDI (023), 8T0 (051), 9ERROR (065), 8T1 (091)

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

705026 9PWRRR, 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) register 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

X and Y = FORTRAN IV REAL entities.

Size: 57

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

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705027 9PWRDD, 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

X and Y = FORTRAN IV DOUBLE PRECISION entities.

Size: 57

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

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705028 9PWRCC, 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 9PWRKK, 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: 9CDLOG (037), 9CDEXP (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 $Z_1 * Z_2$,

where

Z_1 and Z_2 = FORTRAN IV COMPLEX entities.

Size: 10

Subroutines Used: 8T0 (051), 8T1 (091)

(Indirectly): (none)

705032 9CDIV, COMPLEX DIVIDE

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/Z_2,

where

Z_1 and Z_2 = FORTRAN IV COMPLEX entities.

Size: 19

Subroutines Used: 8T0 (051), 8T1 (091)

(Indirectly): (none)

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

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z_1 in register pairs 8-9 and 10-11, Z_2 in register pairs 12-13 and 14-15, register 6 is link, result returned in register pairs 8-9 and 10-11.

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

where

Z_1 and Z_2 = 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: Z_1 in register pairs 8-9 and 10-11, Z_2 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 $Z_1 * Z_2$,

where

Z_1 and Z_2 = FORTRAN IV DOUBLE COMPLEX entities.

Size: 10

Subroutines Used: 8T0 (051), 8T1 (091)

(Indirectly): (none)

705035 9KDIV, DOUBLE COMPLEX DIVIDE

Calling Sequence: Uses FORTRAN IV non-standard calling sequence: Z_1 in register pairs 8-9 and 10-11, Z_2 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 Z_1/Z_2,

where

Z_1 and Z_2 = FORTRAN IV DOUBLE COMPLEX entities.

Size: 17

Subroutines Used: 8T0 (051), 8T1 (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 $\ell n(Z)$,

where

Z = FORTRAN IV COMPLEX entity.

Size: 56

Subroutines Used: 9ALOG (001), 9ATAN1 (007), 9ERROR (065), 8T1 (091)

(Indirectly): 8T0 (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 $\ell n(Z)$, where

where

Z = FORTRAN IV DOUBLE COMPLEX entity.

Size: 53

Subroutines Used: 9DLOG (002), 9DATAN1 (008), 9ERROR (066), 8T1 (091)

(Indirectly): 8T0 (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 = FORTRAN IV COMPLEX entity.

Size: 20

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

(Indirectly): 9EXP (003), 8T0 (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 = FORTRAN IV DOUBLE COMPLEX entity.

Size: 28

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

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705040 9CSIN (9CCOS), (9CSINH), (9CCOSH), COMPLEX CIRCULAR/HYPERBOLIC SINE/COSINE

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 = FORTRAN IV COMPLEX entity.

Size: 86

Subroutines Used: 8T0 (051), 7SIN (052), 9ERROR (065), 8T1 (091)

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

705041 9CDSIN, (9CDCOS), (9CDSINH), (9CDCOSH), DOUBLE COMPLEX CIRCULAR/HYPERBOLIC SINE/COSINE

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 eighteen 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 = FORTRAN IV DOUBLE COMPLEX entity.

Size: 112

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

(Indirectly): 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705042 9CTAN, (9CTANH), COMPLEX CIRCULAR/HYPERBOLIC TANGENT

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, and 11. The reentrant version also uses registers 12 and 13.

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

where

Z = FORTRAN IV COMPLEX entity.

Size: 52

Subroutines Used: 9TANH (013), 9TAN (017), 9ERROR (065), 8T1 (091)

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

705043 9CDTAN, (9CDTANH), DOUBLE COMPLEX CIRCULAR/HYPERBOLIC TANGENT

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 tan(Z) or tanh(Z),

where

Z = FORTRAN IV DOUBLE COMPLEX entity.

Size: 59

Subroutines Used: 9DTANH (014), 9DTAN (018), 9ERROR (065), 8T1 (091)

(Indirectly): 9DEXP (004), 8T0 (051), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705044 9CATAN, COMPLEX ARCTANGENT

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 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)

705045 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)

705046 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 register 4, 12, and 13.

Purpose: Calculates $\sqrt{Z}$ 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)

705047 9CDSQRT, (9CDABS), (7CDSQRT), 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 $\sqrt{Z}$ 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)

705048 9CASIN, (9CACOS), (9CDASIN), (9CDACOS), COMPLEX MULTIPLE PRECISION ARCSINE/ARCCOSINE

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 pairs 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), 8T1 (091)

(Indirectly): 9DLOG (002), 9DATAN1 (008), 9DSQRT (010), 8T0 (051), 9ERROR (065), 7ERROR (066), 8TINIT (092), 8TERROR (093)

705051 8T0, FORTRAN LIBRARY TEMP AREA

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)

705052 7SIN, REAL EXPONENTIAL, SINE, AND COSINE

Calling Sequence: X in register 8, Y in register 9, register 2 is link, e^X returned in register 8, cos(Y) returned in register 12, sin(Y) returned in register 13. Also uses (and does not preserve) registers 10 and 11. The reentrant version also uses register 4 and three words, of which the one with the greatest address may be accessed by an instruction using single-word addressing, a zero address, tagged by register 4.

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), 8T0 (051), 8T1 (091)

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

705053 9IFR, REAL APPROXIMATE EQUALITY TEST

Calling Sequence:
```
(AR0)  =  ARG1
(AR1)  =  ARG2
(AR2)  =  EPSILON
BAL,LL    9IFR
```

Purpose: Accepts three real arguments and returns an integer result in AI.

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:
```
(AD)   =  double precision 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)

705058 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: 8T0 (051), 9DTOR (057)

(Indirectly): 8T1 (091)

705059 9SETUP0, SET UP ZERO ARGUMENTS

Calling Sequence: BAL, LR 9SETUP0

Purpose: Sets up zero arguments in a standard receiving sequence. Essentially, this involves doing nothing. However, 9SETUP0 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)

705060 9SETUP1, SET UP ONE ARGUMENT

Calling Sequence: BAL, LR 9SETUP1
type, P dummy temp

Purpose: Sets up 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)

705061 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)     =  number of dummies
BAL, LR     9SETUPN
type, P     dummy temp 1
type, P     dummy temp 2
  :           :
  :           :
type, P     dummy temp ND
```

for 9SETUPM:

```
(MNA)    =  minimum number of arguments
(MXA)    =  Maximum number of arguments
BAL, LR     9SETUPM
type, P     dummy temp 1
type, P     dummy temp 2
  :           :
  :           :
type, P     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)

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

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 9SETUP0, 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: 14

Subroutines Used: (none)

(Indirectly): (none)

705063 9SETUPV, SET UP A VARIABLE NUMBER OF ARGUMENTS (MULTIPLE DUMMY)

```
Calling Sequence:  (ND)     =  - number of dummies (including multiple dummy)
                   BAL, LR     9SETUPV
                   type, P     fixed dummy temp 1
                   type, P     fixed dummy temp 2
                   :           :
                   type, P     fixed dummy temp (number of dummies (negative))
                   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 (LR) - (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 means 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)

705064 9INITIAL, RUN-TIME INITIALIZATION

Calling Sequence: 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 executably 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)

705065 9ERROR, MATH LIBRARY ERROR REPORTING

Calling Sequence:

```
(LL)                =  exit from math routine
BAL, LE                9ERROR
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 nonpositive 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), 8TERROR (093)

(Indirectly): 8TINIT (092)

Calling Sequence:

For 7ERROR:
```
EL      = entry location
EN      = address of the two-word entry name
BL      = BA (TEXTC '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(TEXTC '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 (TEXTC '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 (TEXTC '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 proclaiming a FORTRAN run-time error.

7ERRTEXT: Prints error message and determines whether to abort the job or return, depending on the error severity. (A severity level of 15 is always sufficient to cause an abort.)

7ERRMARK: Prints all or part of the specified error message buffer followed by a line containing a vertical bar beneath the erroneous character.

7ERRINIT: Initializes the error message buffer to the empty state.

7PRC: Inserts the given character into the error message buffer.

7PRQ: Inserts the given quote string into the error message buffer.

7PAC: Inserts the string into the error message buffer.

7PHC: Converts hexadecimal value into EBCDIC characters and inserts them into the error message buffer.

7PRL: Prints the error message buffer on the DO device, and then reinitializes the error message buffer.

7BUFOUT: Prints the string on the DO device.

7BUFOUTC: Prints the quote string on the DO device.

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)

705067 9BCDREAD (9READ) (7BCDREAD), BCD READ

Calling Sequence:
```
(AI)     =  unit number
(FP)     =  word address of FORMAT
BAL, LL     9BCDREAD
```

Purpose:

9BCDREAD: BCD READ subroutine for such statements as

```
READ(unit,FORMAT) LIST
```
or
```
READ(105,FORMAT) list
```

9READ: Special entry point for card reader, e.g.,

```
READ FORMAT,list
```
or
```
READ(105,FORMAT) list
```

7BCDREAD: Special entry point used by 9BCDRDEE, which processes ERR= and END=, e.g.,

```
READ(unit,FORMAT,END=3,ERR=4) list
```

This routine sets up the format scan routine (9IEDIT) to interface directly with the user program in obtaining locations of list items.

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

9BCDREAD provides 9IEDIT 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, 9BCDRDEE assumes control.

Size: 48

Subroutines Used: 9IEDIT (072), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092), 8TERROR (093), 8TEDIT (094)

(Indirectly): 8TO (051), 7ERROR (066), 9IODATA (074), 9STOP (088), 9BINDEC (089), 7GETMODE (090)

705068 9BCDWRIT (9PRINT), BCD WRITE

Calling Sequence:
```
(AI)     =  unit number
(FP)     =  word address of FORMAT
BAL, LL     9BCDWRIT
```

Purpose:

9BCDWRIT: BCD WRITE subroutine for such statements as WRITE(unit,FORMAT)list

9PRINT: Special entry point for line printer, e.g.,

```
PRINT FORMAT,list
```
or
```
WRITE(108,FORMAT) list
```

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

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

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.

705068 9BCDWRIT (9PRINT), BCD WRITE (cont.)

Size: 57

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

(Indirectly): 8T0 (051), 7ERROR (066), 9IODATA (074), 7BINDEC (089), 7GETMODE (090), 8TINIT (092)

705069 9BINREAD (9BINWRIT) (7BINREAD), BINARY READ AND WRITE

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

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:

Word	Byte	Contents
0		First control word
	0	X'3C' = Not last physical record X'1C' = Last physical record
	1	Byte checksum
	2 + 3	Number of data bytes in record + 8 (i.e., includes control words)

705069 9BINREAD (9BINWRIT) (7BINREAD), BINARY READ AND WRITE (cont.)

Word	Byte	Contents
1-N		Data words (may be none)
N + 1		Second control word
	0	Same as word 0, byte 0
	1	X'BD' (special binary code)
	2 + 3	Physical record number (starts at zero)

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: 8T0(051), 7ERROR (066), 9IODATA (074), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092), 8TERROR (093), 8TEDIT (094)

(Indirectly): 9STOP (088), 7BINDEC (089), 7GETMODE (090)

705070 9DECODE (9ENCODE), MEMORY-TO-MEMORY DATA CONVERSION

Calling Sequence: (AI) = number of characters per internal record
(FP) = starting location of format (word address)
(X5) = starting location of internal buffer (words)
BAL, LL 9DECODE/9ENCODE

On return from 9ENDIOL:

(AI) = number of characters processed

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

Like 9BCDREAD/9BCDWRIT, 9DECODE/9ENCODE provide 9IEDIT/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), 9IEDIT (072), 8TERROR (093), 8TEDIT (094)

(Indirectly): 8T0 (051), 9IODATA (074), 7BINDEC (089), 7GETMODE (090)

705073 9IOLUSA, I/O LIST UNSUBSCRIPTED ARRAY TRANSMITTER

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)

705074 9IODATA(9DATA) (9ENDIOL), 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 8IODADDR, the datum type (as a code between 1 and 6) in bits 1 through 31 of 8IODTYPE, and the protection bit in bit 0 of 8IODTYPE.

9DATA sets '8IOTRIG' true to allow 9IODATA and 9ENDIOL to receive calls.

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

Size: 132

Subroutines Used: 7ERROR (066), 7GETMODE (090), 8TERROR (093), 8TEDIT (094)

(Indirectly): 8T0 (051), 8TINIT (092)

705075 9REWIND, REWIND SEQUENTIAL FILE

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

Purpose: Rewinds any sequential file.

Size: 14

Subroutines Used: 7UNITADR (080)

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

705076 9BKSPACE, BACKSPACE ONE LOGICAL RECORD

Calling Sequence: (AI) = 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:

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

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: 8T0 (051), 7EOFABRT (078), 7UNITADR (080), 8TINIT (092), 8TERROR (093), 8TEDIT (094)

(Indirectly): 7ERROR (066), 9STOP (088), 7BINDEC (089)

705077 9ENDFILE, WRITE END-OF-FILE

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

Purpose: Writes end-of-file

Size: 14

Subroutines Used: 7UNITADR (080), 8TERROR (093), 8TEDIT (094)

(Indirectly): 8T0 (051), 7ERROR (066), 7BINDEC (089), 8TINIT (092)

705078 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): 8T0 (051), 7BINDEC (089), 8TINIT (092)

705080 7UNITADR, FIND I/O UNIT DCB ADDRESS

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 maximum permissible number is 9999) and stored in 8UNITVAL. 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 8UNITNAM 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 6DCBNAME preceded by the two characters "F:". For example, at this point the unit number 108 would appear in 8UNITNAM as 03F1F0F8 and 40404040, while in 6DCBNAME it would be C67AF1F0 and F8404040. 6DCBNAME 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 8DCBADR 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), 8TERROR (093), 8TEDIT (094)

(Indirectly): 8T0 (051), 8TINIT (092)

705081 9ASFORM, ASSIGNED FORMAT

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)

705082 9ASGOTO, ASSIGNED GO TO

Calling Sequence: (X7) = 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), 8TERROR (093)

(Indirectly): 8TINIT (092)

705083 9IFSWICH, TEST SENSE SWITCH

Calling Sequence:

(AI)	=	switch number (integer value)
BAL, LL		9IFSWICH
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,

!SWITCH 0000,(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), 8TERROR (093)

705084 9SNSLITE (9IFSLITE), SENSE LIGHT SET AND TEST

Calling Sequence:

For 9SNSLITE:

(AI)	=	sense light number
BAL, LL		9SNSLITE

For 9IFSLITE:

(AI)	=	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): 8TERROR (093)

705085 9IFOVFL, FLOATING OVERFLOW TEST

Calling Sequence: BAL, LL 9IFOVFL
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 (9IEDIT). 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) = positive decimal integer
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): 8T0 (051)

705088 9STOP (7STOP), STOP

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

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): 8T0 (051)

705089 7BINDEC, BINARY TO DECIMAL FOR MESSAGES

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

Purpose: Converts the value in AI 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: 8T0 (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 representing 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: 8T0 (051)

(Indirectly): (none)

705091 8T1, 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 8T0+1, 8T0+2, etc. However, there are still numerous routines in the library that reference them as 8T0, 8T1, 8T2, etc., thus requiring the loading of this program. After the entire library has been converted to reference only 8T0, this program will be removed.

This defines the names 8T1 through 8T17.

Size: 1

Subroutines Used: 8T0 (051)

(Indirectly): (none)

705092 8TINIT, 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 8T0.)

Size: 14

Subroutines Used: (none)

(Indirectly): (none)

705093 8TERROR, TEMPS FOR 7ERROR

Calling Sequence: N/A

Purpose: Contains the temps and special storage areas used by the run-time error routine (7ERROR). (For other temps see 8TEDIT, 8TINIT, and 8T0.)

Size: 52

Subroutines Used: (none)

(Indirectly): (none)

705094 8TEDIT, 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 8T0.)

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 expontial (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), 8T0 (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), 8T0 (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                 LI, NA    2
BAL, LC   name      or      BAL, LC   name
SNGL      ARG               SNGL      ARG1
                            SNGL      ARG2
```

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

Size: 11

Subroutines Used: 9ATAN1 (007), 9SETUPN (062)

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

705108 DATAN, DRIVER FOR 9DATAN1, 9DATAN2

Calling Sequence:
```
LI, NA    1                 LI, NA    2
BAL, LC   name      or      BAL, LC   name
DOUB      ARG               DOUB      ARG1
                            DOUB      ARG2
```

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

Size: 11

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

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

705109 SQRT, DRIVER FOR 9SQRT

Calling Sequence:
```
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:
```
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), 0SETUP1 (060)

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

705111 SINH, DRIVER FOR 9SINH

Calling Sequence:
```
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:
```
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), 9SETUP1 (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), 9SETUP1 (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: 9ASIN (015), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9ATAN1 (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), 9SETUP1 (060)

(Indirectly): 9DATAN1 (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), 8T0 (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), 8T0 (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), 8T0 (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), 8T0 (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    1
BAL, LC   name
SNGL      ARG
```

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

Size: 5

Subroutines Used: 9ASIN (015), 8T0 (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    1                 LI, NA    2
BAL, LC   name      or      BAL, LC   name
SNGL      ARG               SNGL      ARG1
                            SNGL      ARG2
```

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.

Subroutines Used: 9ATAN1 (007), 8T0 (051), 9SETUPN (062)

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

705123 COS, DRIVER FOR 9COS

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

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

Size: 5

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

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

705124 COSH, DRIVER FOR 9COSH

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

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

Size: 5

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

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

705126 DATAN2, DRIVER FOR 9DATAN1, 9DATAN2

Calling Sequence:
```
LI, NA    1              LI, NA    2
BAL, LC   name     or     BAL, LC   name
DOUB      ARG             DOUB      ARG1
                          DOUB      ARG2
```

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), 9SETUPN (062)

(Indirectly): 9ERROR (065), 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), 9SETUP1 (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), 9SETUP1 (060)

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

705129 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), 7SIN (052), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705130 CDCOS, DRIVER FOR 9CDCOS

Calling Sequence:
```
LI, NA    1
BAL, LC   name
KMPX      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)

705131 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)

705132 CDTANH, DRIVER FOR 9CDTANH

Calling Sequence:
```
LI, NA    1
BAL, LC   name
KMPX      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:
```
LI, 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:
```
LI, 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 CDACOS, DRIVER FOR 9CDACOS

Calling Sequence:
```
LI, 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:
```
LI, 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), 9ATAN1 (007), 9ERROR (065), 7ERROR (066), 8T1 (091) 8TINIT (092), 8TERROR (093)

705137 CDLOG, DRIVER FOR 9CDLOG

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

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

Size: 7

Subroutines Used: 9CDLOG (037), 8T0 (051), 9SETUP1 (060)

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

705138 CEXP, DRIVER FOR 9CEXP

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

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

Size: 5

Subroutines Used: 9CEXP (038), 8T0 (051), 9SETUP1 (060)

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

705139 CDEXP, DRIVER FOR 9CDEXP

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

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

Size: 7

Subroutines Used: 9CDEXP (039), 8T0 (051), 9SETUP1 (060)

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

705140 CSIN, DRIVER FOR 9CSIN

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

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

Size: 5

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

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

705142 CTAN, DRIVER FOR 9CTAN

Calling Sequence:
```
LI, 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), 8T0 (051), 9SETUP1 (060)

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

705143 CDTAN, DRIVER FOR 9CDTAN

Calling Sequence:
```
LI, 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), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9DEXP (003), 9DTANH (014), 9DTAN (018), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705144 CATAN, DRIVER FOR 9CATAN

Calling Sequence:
```
LI, 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), 8T0 (051), 9SETUP1 (060)

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

705145 CDATAN, DRIVER FOR 9CDATAN

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

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

Size: 7

Subroutines Used: 9CDATAN (045), 8T0 (051), 9SETUP1 (060)

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

705146 CSQRT, DRIVER FOR 9CSQRT

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

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

Size: 5

Subroutines Used: 9CSQRT (046), 8T0 (051), 9SETUP1 (060)

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

705147 CDSQRT, DRIVER FOR 9CDSQRT

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

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

Size: 7

Subroutines Used: 9CDSQRT (047), 8T0 (051), 9SETUP1 (060)

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

705148 CASIN, DRIVER FOR 9CASIN

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

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

Size: 5

Subroutines Used: 9CASIN (048), 8T0 (051), 9SETUP1 (060)

(Indirectly): 9DLOG (002), 9DSQRT (010), 9DATAN1 (008), 9CDLOG (037) 9CDSQRT (047), 9ERROR (065), 7ERROR (066), 8T1 (091), 8TINIT (092), 8TERROR (093)

705149 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), 8T0 (051), 9SETUP1 (060)

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

705150 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), 8T0 (051), 9SETUP1 (060)

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

705151 CDSINH, DRIVER FOR 9CDSINH

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

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

Size: 7

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

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

705152 CDCOSH, DRIVER FOR 9CDCOSH

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

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

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: 8T0 (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: 8T0 (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: 8T0 (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: 8T0 (051), 9DTOI (056), 9SETUP1 (060)

(Indirectly): (none)

705157 SNGL, DRIVER FOR 9DTOR

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

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

Size: 5

Subroutines Used: 8T0 (051), 9DTOR (057), 9SETUP1 (060)

(Indirectly): (none)

705158 CSNGL, DRIVER FOR 9KTOC

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

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

Size: 5

Subroutines Used: 8T0 (051), 9KTOC (058), 9SETUP1 (060)

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

705159 CABS, DRIVER FOR 9CABS

Calling Sequence:
```
LI, NA    1
BAL, 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), 8T0 (051), 9SETUP1 (060)

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

705160 CDABS, DRIVER FOR 9CDABS

Calling Sequence:
```
LI, NA    1
BAL, 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), 8T0 (051), 9SETUP1 (060)

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

705161 ACOSF, DRIVER FOR 9ACOS

Calling Sequence:
```
LI, NA    1
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), 9SETUP1 (060)

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

705162 ARCOS, DRIVER FOR 9ACOS

Calling Sequence:
```
LI, NA    1
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), 9SETUP1 (060)

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

705163 ASINF, DRIVER FOR 9ASIN

Calling Sequence:
```
LI, NA    1
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), 9SETUP1 (060)

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

705164 ARSIN, DRIVER FOR 9ASIN

Calling Sequence:
```
LI, NA    1
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), 9SETUP1 (060)

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

705165 ATANF, DRIVER FOR 9ATAN1, 9ATAN2

Calling Sequence:

LI, NA	1		LI, NA	2
BAL, LC	name	or	BAL, LC	name
SNGL	ARG		SNGL	ARG_1
			SNGL	ARG_2

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: 9ATAN1 (007), 8T0 (051), 9SETUPN (062)

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

705166 COSF, DRIVER FOR 9COS

Calling Sequence:

LI, 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), 8T0 (051), 9SETUP1 (060)

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

705167 COSHF, DRIVER FOR 9COSH

Calling Sequence:

LI, 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), 8T0 (051), 9SETUP1 (060)

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

705168 DARCOS, DRIVER FOR 9DACOS

Calling Sequence:

LI, 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), 8T0 (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 (705116), provided for compatibility with IBM 360.

Size: 5

Subroutines Used: 9DASIN (016), 8T0 (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), 8T0 (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: 8T0, (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: 8T0 (051), 9DTOI (056), 9SETUP1 (060)

(Indirectly): (none)

705173 LOG, DRIVER FOR 9ALOG

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

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:
```
LI, NA    1
BAL, LC   name
SNGL      ARG
```

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:
```
LI, NA    1
BAL, LC   name
SNGL      ARG
```

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:
```
LI, NA    1
BAL, LC   name
SNGL      ARG
```

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)

705181 AIMAG, REAL IMAGINARY PART

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

Purpose: Imaginary part of complex argument (basic external version).

Size: 4

Subroutines Used: 9SETUP1 (060)

(Indirectly): (none)

705182 AINT, REAL INTEGRAL VALUE

Calling Sequence:
```
LI, 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)

705183 AMAX, REAL MAXIMUM VALUE

Calling Sequence:
```
LI, 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)

705184 AMAX1, REAL MAXIMUM VALUE

Calling Sequence:
```
LI, 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 AMAX0, REAL MAXIMUM VALUE OF INTEGERS

Calling Sequence:

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

Purpose: Conversion of greatest integer argument to real value (basic external version).

Size: 17

Subroutines Used: 8T0 (051), 9ITOD (055), 9SETUPV (063)

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

705186 AMIN, REAL MINIMUM VALUE

Calling Sequence:

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

Purpose: Smallest argument (basic external version).

Size: 16

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

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

705187 AMIN1, REAL MINIMUM VALUE

Calling Sequence:

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

Purpose: This is an alternate version of AMIN (705186), provided for compatibility with other FORTRAN systems.

Size: 16

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

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

705188 AMIN0, REAL MINIMUM VALUE OF INTEGERS

Calling Sequence:

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

Purpose: Conversion of smallest integer argument to real value (basic external version).

Size: 17

Subroutines Used: 8T0 (051), 9ITOD (055), 9SETUPV (063)

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

705189 AMOD, REAL REMAINDER (MODULO)

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

Purpose: Arg_1 (mod arg_2): evaluated as $arg_1 - arg_2 * AINT(arg_1/arg_2)$, i.e., the sign is the same as arg_1. Function is undefined if $arg_2 = 0$ (basic external version).

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

(Indirectly): 8T1 (091)

705190 CDBLE, COMPLEX TO DOUBLE COMPLEX

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

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

Size: 7

Subroutines Used: 9SETUP1 (060)

(Indirectly): (none)

705191 CDINT, DOUBLE COMPLEX INTEGRAL VALUE

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

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

(Indirectly): (none)

705192 CINT, COMPLEX INTEGRAL VALUE

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

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

(Indirectly): (none)

705193 CMPLX, COMPLEX FROM TWO REALS

Calling Sequence:
```
LI, NA    2
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: 8T0 (051), 9SETUP2 (061)

(Indirectly): 8T1 (091)

705194 CONJG, COMPLEX CONJUGATE

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

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

Size: 4

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

(Indirectly): (none)

705196 DBLE, REAL TO DOUBLE CONVERSION

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

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

Size: 5

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

(Indirectly): 8T1 (091)

705198 DCONJG, DOUBLE COMPLEX CONJUGATE

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

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

Size: 6

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

(Indirectly): 8T1 (091)

705201 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)

705202 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)

705203 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)

705204 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        ARG1
DOUB        ARG2
:           :
:           :
DOUB        ARGn
```

Purpose: Smallest argument (basic external version).

Size: 19

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

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

705206 DMIN1, DOUBLE PRECISION MINIMUM VALUE

Calling Sequence:

```
LI, NA      n
BAL, LC     (name)
DOUB        ARG1
DOUB        ARG2
:           :
:           :
DOUB        ARGn
```

Purpose: This is an alternate version of DMIN (705205), provided for compatibility with other FORTRAN systems.

Size: 19

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

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

705207 DMOD, DOUBLE PRECISION REMAINDER (MODULO)

Calling Sequence:

```
LI, NA      2
BAL, LC     name
DOUB        ARG1
DOUB        ARG2
```

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

(Indirectly): (none)

705209 DSIGN, DOUBLE PRECISION TRANSFER OF SIGN

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

Purpose: Magnitude of arg_1 with sign of arg_2 (zero is considered positive) (basic external version).

Size: 8

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

(Indirectly): 8T1 (091)

705210 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: 8T0 (051), 9SETUP1 (060)

(Indirectly): (none)

705211 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: 8T0 (051), 9SETUPV (063)

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

705212 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: 8T0 (051), 9SETUP1 (060)

(Indirectly): (none)

705213 IDIM, INTEGER POSITIVE DIFFERENCE

Calling Sequence:

```
LI, NA    2
BAL, LC   name
INTG      ARG1
INTG      ARG2
```

Purpose: Integer positive difference: $IDIM(x,y) = x_1 - \min(x_1, y_2)$

Size: 8

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

(Indirectly): 8T1 (091)

705214 IEOR, INTEGER BOOLEAN EXCLUSIVE OR

Calling Sequence:

```
LI, NA    n
BAL, LC   (name)
INTG      ARG1
INTG      ARG2
.         .
.         .
INTG      ARGn
```

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

Size: 14

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

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

705215 IEXCLR, INTEGER EXCLUSIVE OR

Calling Sequence:

```
LI, NA    n
BAL, LC   (name)
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: 8T0 (051), 9SETUPV (063)

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

705216 IF, DRIVER FOR 9IFR

Calling Sequence:

```
LI, NA    1              LI, NA    2              LI, NA    n
BAL, LC   name or        BAL, LC   name or        BAL, LC   (name)
SNGL      ARG            SNGL      ARG1           SNGL      ARG1
                         SNGL      ARG2           SNGL      ARG2
                                                  SNGL      ARG3
```

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: 8T0 (051), 9IFR (053), 9SETUPN (062)

(Indirectly): (none)

705217 INOT, 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)

705218 IOR, 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)

705219 ISIGN, INTEGER TRANSFER OF SIGN

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

Purpose: Magnitude of arg_1 with sign of arg_2 (zero is considered positive) (basic external version).

Size: 8

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

(Indirectly): 8T1 (091)

705220 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      n
BAL, LC     (name)
INTG        ARG1
INTG        ARG2
 .           .
 .           .
INTG        ARGn
```

Purpose: Greatest argument (basic external version).

Size: 16

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

(Indirectly): 9SETUPN (062)

705222 MAX0, INTEGER MAXIMUM VALUE

Calling Sequence:

```
LI, NA      n
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: 8T0 (051), 9SETUPV (063)

(Indirectly): 9SETUPN (062)

705223 MAX1, INTEGER MAXIMUM VALUE OF REALS

Calling Sequence:

```
LI, NA      n
BAL, LC     (name)
SNGL        ARG1
SNGL        ARG2
 .           .
 .           .
SNGL        ARGn
```

Purpose: Conversion of greatest real argument (basic external version).

Size: 17

Subroutines Used: 8T0 (051), 9DTOI (056), 9SETUPV (063)

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

705224 MIN, INTEGER MINIMUM VALUE

Calling Sequence:

```
LI, NA      n
BAL, LC     (name)
INTG        ARG1
INTG        ARG2
 .           .
 .           .
INTG        ARGn
```

Purpose: Smallest argument (basic external version).

Size: 16

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

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

705225 MIN0, INTEGER MINIMUM VALUE

Calling Sequence:

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

Purpose: This is an alternate version of MIN (705224), provided for compatibility with other FORTRAN systems.

Size: 16

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

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

705226 MIN1, INTEGER MINIMUM VALUE OF REALS

Calling Sequence:

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

Purpose: Conversion of smallest real argument (basic external version).

Size: 17

Subroutines Used: 8T0 (051), 9DTOI (056), 9SETUPV (063)

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

705227 MOD, INTEGER REMAINDER (MODULO)

Calling Sequence:

	LI, NA	2
	BAL, LC	name
	INTG	ARG_1
	INTG	ARG_2

Purpose: Arg_1 (mod arg_2): evaluated as $arg_1 - arg_2 * INT(arg_1/arg_2)$, i.e., the sign is the same as arg_1. Function is undefined if $arg_2 = 0$ (basic external version).

Size: 8

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

(Indirectly): 8T1 (091)

705228 REAL, REAL PART OF COMPLEX

Calling Sequence:

	LI, NA	1
	BAL, LC	name
	CMPX	ARG

Purpose: Real part of complex argument (basic external version).

Size: 4

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

(Indirectly): (none)

705229 SIGN, REAL TRANSFER OF SIGN

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

Purpose: Magnitude of arg_1 with sign of arg_2 (zero is considered positive) (basic external version).

Size: 8

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

(Indirectly): 8T1 (091)

705230 ABSF, REAL ABSOLUTE VALUE

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

Purpose: This is an alternate version of ABS (705180), provided for compatibility with FORTRAN II.

Size: 4

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

(Indirectly): (none)

705231 DIMF, REAL POSITIVE DIFFERENCE

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

Purpose: This is an alternate version of DIM (705200), provided for compatibility with FORTRAN II.

Size: 8

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

(Indirectly): 8T1 (091)

705232 SIGNF, REAL MAGNITUDE

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

Purpose: This is an alternate version of SIGN (705229), provided for compatibility with FORTRAN II.

Size: 8

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

(Indirectly): 8T1 (091)

705233 SSWTCH, SENSE SWITCH TEST

```
Calling Sequence:  LI,NA    1                  LI,NA    2
                   BAL,LC   SSWTCH     or     BAL,LC   SSWTCH
                   INTG     N                 INTG     N
                                              INTG     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. SSWTCH 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 9IFSWICH does the real work.

For further information on the use of sense switches, see 9IFSWICH (083).

Size: 17

Subroutines Used: 8T0 (051), 9SETUPN (062), 9IFSWICH (083)

(Indirectly): 7ERROR (066)

705234 SLITET, SENSE LIGHT TEST

```
Calling Sequence:  LI,NA    1                  LI,NA    2
                   BAL,LC   SLITET     or     BAL,LC   SLITET
                   INTG     N                 INTG     N
                                              INTG     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: 8T0 (051), 9SETUPN (062), 9SNSLITE (084)

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

705235 SLITE, SET SENSE LIGHT

```
Calling Sequence:  LI,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)

705236 OVERFL, TEST FOR FLOATING OVERFLOW

```
Calling Sequence:  LI,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)

705237 DVCHK, TEST FOR FLOATING OVERFLOW (DIVIDE CHECK)

```
Calling Sequence:  LI,NA     1
                   BAL,LC    DVCHK
                   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: Performs exactly the same test as OVERFL. DVCHK is included in the library only for compatibility with systems that use a Divide Check Indicator, as in IBM 7090 systems. DVCHK 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. After the test, the overflow trigger 8FLOVTRG is turned off.

For further information on floating overflow see 9IFOVFL (085)

Size: 8

Subroutines Used: 9SETUP1 (060), 9IFOVFL (085)

(Indirectly): 8TINIT (092)

705238 EXIT, EXIT TO THE MONITOR

```
Calling Sequence:  LI,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:
```
LI,NA    2
BAL,LC   EOFSET
INTG     LOC
INTG     UNIT
```

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:
```
LI,NA    2
BAL,LC   SETEOF
INTG     UNIT
INTG     LOC
```

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: 9SETUPN (062), 7ERROR (066), 7EOFABRT (078), 8TINIT (092), 8TEDIT (094)

(Indirectly): 7ERROR (066), 9STOP (088), 8TERROR (093)

705241 BUFFERIN, DIRECT INPUT

```
Calling Sequence:  LI,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 MODE = 0, the mode is BCD(EBCDIC); if MODE ≠ 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: 8T0 (051), 9SETUPN (062), 7UNITADR (080), 8TEDIT (094)

(Indirectly): 7ERROR (066), 7BINDEC (089), 8TERROR (093)

705242 BUFFEROU, DIRECT OUTPUT

```
Calling Sequence:  LI,NA     6
                   BAL,LC    BUFFERIN
                   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: 8T0 (051), 9SETUPN (062), 7UNITADR (080), 8TEDIT (094)

(Indirectly): 7ERROR (066), 7BINDEC (089), 8TERROR (093)

705243 ABORTSET, SET UP ABORT EXIT AND SEVERITY LEVEL

Calling Sequence:

```
LI,NA     2
BAL,LC    ABORTSET
INTG      LOC
INTG      LEVEL
```

where

LOC is a statement number (or assigned variable) to which 7ERROR will transfer when an abort level error occurs. A value of zero (instead of a statement number) resets 7ERROR to abort to the Monitor.

LEVEL is an optional integer value from 1 through 15. 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 ABORTSET is called with only the LOC argument, the abort exit (8ABORTEX) is set up, but the abort severity (8ABRTSEV) is left alone.

Purpose: 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: 8T0 (051), 9SETUPN (062), 8TINIT (092)

(Indirectly): (none)