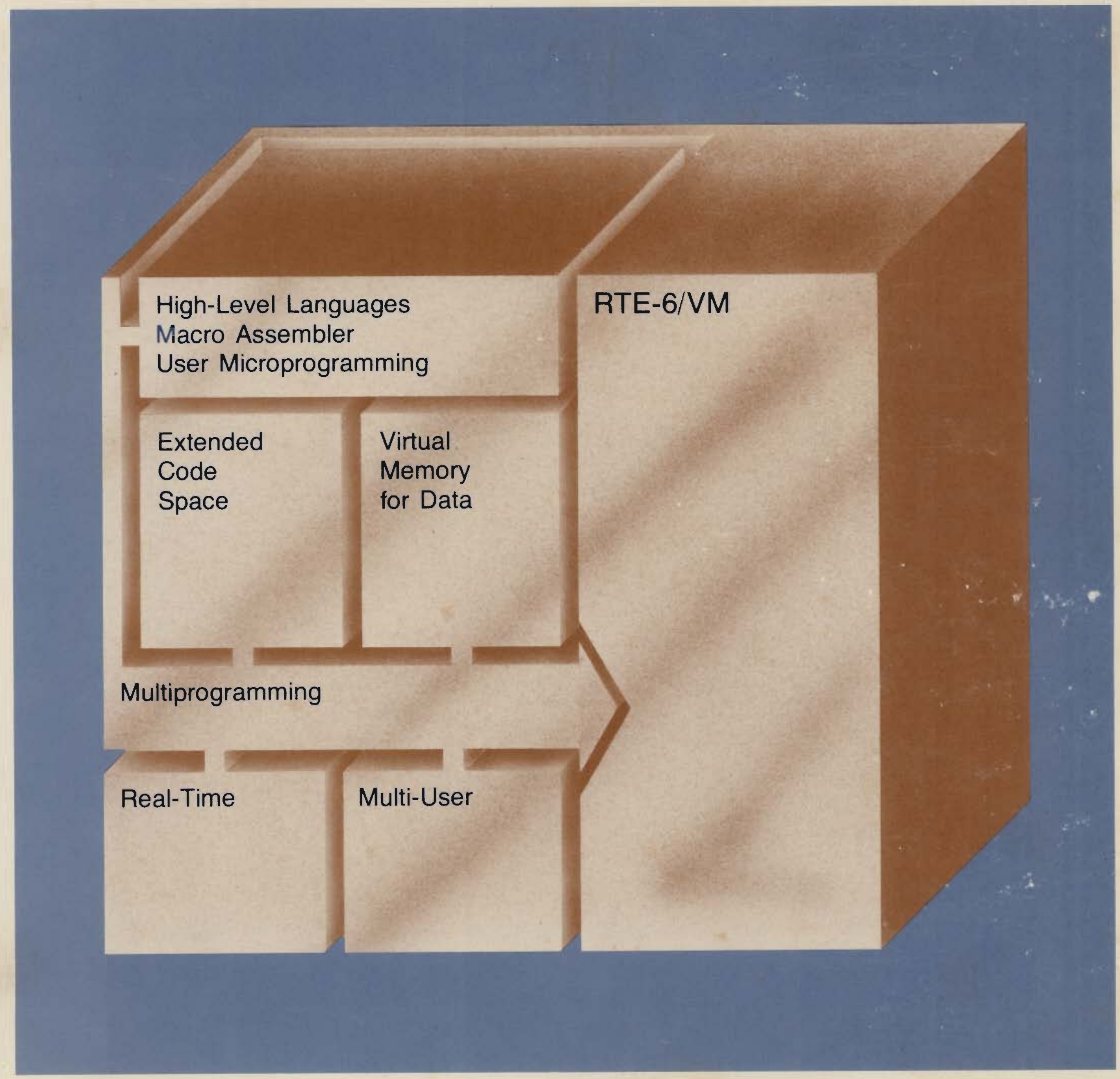
HEWLETT PACKARD

RTE-6/VM Relocatable Library

Reference Manual



RTE-6/VM

Relocatable Library

Reference Manual

HEWLETT
PACKARD

HEWLETT-PACKARD COMPANY
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014

MANUAL PART NO. 92084-90013
Printed in U.S.A. December 1981
U0782

PRINTING HISTORY

The Printing History below identifies the Edition of this Manual and any Updates that are included. Periodically, Update packages are distributed which contain replacement pages to be merged into the manual, including an updated copy of this Printing History page. Also, the update may contain write-in instructions.

Each reprinting of this manual will incorporate all past Updates, however, no new information will be added. Thus, the reprinted copy will be identical in content to prior printings of the same edition with its user-inserted update information. New editions of this manual will contain new information, as well as all Updates.

To determine what software manual edition and update is compatible with your current software revision code, refer to the appropriate Software Numbering Catalog, Software Product Catalog, or Diagnostic Configurator Manual.

First Edition .... Dec 1981
Update 1 .... Jul 1982

Preface

This manual is a programmer's guide to subroutines contained in RTE-6/VM Operating System and describes the following libraries:

$MLIB1	Math/Formatter Library, Part 1
$MLIB2	Math/Formatter Library, Part 2
$6SYLB	RTE-6/VM System Library

Chapter 1 introduces the libraries and explains the format used by this manual to describe the individual subroutines.

Chapter 2 through Chapter 5, are alphabetical groupings of the mathematical, double-integer, utility and library subroutines.

Chapter 6 is a list of run time error messages.

Appendix A identifies the library subroutines that can be called by FORTRAN 4X, FORTRAN 77, and Pascal.

Another relocatable library currently distributed with the RTE-6/VM Operating System is:

%DECAR   Decimal String Subroutine

These subroutines are described in the Decimal String Arithmetic Routines manual (part no. 02100-90140).

Other collections of HP relocatable subroutines for more general use are grouped into other libraries distributed with the RTE-6/VM Operating System. In addition, many RTE subsystems and languages, such as Pascal/1000 and Spooling, include subroutines that may be of general use. Refer to appropriate subsystem and language manuals for more information.

Table of Contents

Chapter 1 Introducing the Libraries

Chapter 2 Mathematical Subroutines

Chapter 1
Introducing the Libraries

Introduction

RTE-6/VM operating systems are delivered with a collection of relocatable subroutines that comprise the system and math/formatter. This group of subroutines is specific to RTE-6/VM operating systems and is used to interface user programs with system services.

Calling Library Subroutines

Library subroutines are called by user programs and are linked to the caller either at generation or load time. These subroutines can be called either by disc-resident or memory-resident programs.

Subroutines referenced by disc-resident programs are appended to the end of the calling program and linked to it either by the loader (LOADR or MLLDR) or the On-Line Generator.

During generation, subroutines referenced by memory-resident programs will be placed in the memory-resident library. These subroutines must either be reentrant or privileged. Several memory-resident programs can then share one subroutine, which can save considerable space in the memory-resident area. Disc-resident programs cannot access routines in the memory-resident library, therefore, copies of these subroutines are created by the generator, placed in the disc-resident library, and appended the disc-resident programs.

If only one memory-resident program is to access a subroutine, it is advantageous to make it a Type 7 subroutine to force it to be appended to the calling program. A Type 7 subroutine is not placed in the memory-resident library and therefore need not be privileged or reentrant. This results in faster execution, since the subroutine will not incur the overhead associated with reentrant or privileged subroutines.

Reentrant Subroutine Structure

A subroutine must meet two criteria to be reentrant:

1. It must not modify any of its own instructions.
2. It must save all temporary results if it is to be called again before completing its current task.

A subroutine saves temporary results in a Temporary Data Buffer (TDB) that the operating system ensures is unique to each program. For example, assume PROGA is executing a reentrant subroutine that is interrupted by PROGB. If PROGB then begins execution of the same subroutine, the system saves PROGA's variables in a TDB in System Available Memory (SAM) until PROGA resumes execution, at which time it restores the proper TDB.

Each time a reentrant subroutine begins executing, the address and length of its temporary data block are transferred to the operating system through the entry point $LIBX in order to save the data. At the end of execution, the reentrant subroutine again calls RTE-6/VM through entry point $LIBX to restore any previous temporary data.

The reentrant subroutine structure is used for subroutines with an execution time exceeding one milli-second. However, for shorter execution times, the overhead time the system uses in saving and restoring temporary data makes reentrant structure unreasonable. Faster subroutines can be structured as privileged.

NOTE

> A library (Type 6) subroutine can only call another library subroutine or Table Area I or Table Area II entry points.

The format and calling sequence for reentrant subroutines is as follows:

```
        NAM   xxxxx,6
        EXT   $LIBR,$LIBX
 ENTRY  NOP                   Entry point of subroutine
        JSB   $LIBR           Tell system to go reentrant
        DEF   TDB             Address of temporary data
          .
          .                   Subroutine instructions go here
          .
 EXIT   JSB   $LIBX           Tell system reentrant run is finished
        DEF   TDB             Address of temporary data
        DEC   N               Return adjustment
                                 (Return point=N+ENTRY)

 TDB    NOP                   System-supplied link to previous TDB
        DEC   K               Total length of current TDB in words
        NOP                   System-returned  address to calling
                                 program

        BSS   K-3
          -
          -
          -                   Temporary data (K-3 words)
          -
```

Privileged Subroutine Structure

Privileged subroutines execute with the interrupt system turned off. This feature allows many memory-resident programs to use a single privileged subroutine without incurring reentrant overhead. As a result, privileged subroutines need not save temporary data buffers but must execute very rapidly (<1 mec) to minimize the time that the interrupt system is disabled.

Since privileged subroutines disable the interrupt system, EXEC calls are illegal within a privileged subroutine. If one is attempted, the calling program will be aborted with an EX error.

The format and calling sequence for privileged subroutines is as follows:

```
       NAM  xxxx,6
       EXT  $LIBR,$LIBX
 ENTRY NOP                  Entry point to the routine
       JSB  $LIBR           Call the system to disable the
                              interrupt system and memory
                              protect fence
       NOP                  Denotes privileged format
        .
        .                   Subroutine instructions
        .
 EXIT  JSB  $LIBX           Call the system to return to calling
                              program, and to enable interrupts
                              and memory protect fence
 EXIT1 DEF  ENTRY           Return address
```

It is also possible to go privileged in a block of in-line code, as follows:

```
        -
        -
       JSB  $LIBR           Go privileged
       NOP                  Denotes privileged format
        -                   First instruction
        -
        -
        -
       JSB  $LIBX           Leave privileged status
       DEF  *+1             Both DEF's are required
       DEF  *+1
```

For greater flexibility, subroutines can be treated as reentrant or privileged when they are generated into the system or as ordinary routines if they are relocated with LOADR or MLLDR. Two library routines, .ZPRV for privileged subroutines, and .ZRTN for reentrant subroutines, cause subroutine code to be modified when it is relocated by the generator.

For subroutines that call .ZRTN, the generator will modify the code as follows:

```
Original Code          Modified at Generator Relocatio

ENTRY NOP              ENTRY NOP
      JSB .ZRTN              JSB $LIBR
      DEF EXIT               DEF TDB
       .               Body of subroutine
       .                      .
       .                      .
EXIT  ENTRY,I          EXIT  JSB $LIBX
      DEF TDB                DEF TDB
      DEC 0                  DEC 0
```

For subroutines that call .ZPRV, the generator will modify the code as follows:

```
Original Code          Modified at Generator Relocatio

ENTRY NOP              ENTRY NOP
      JSB .ZPRV              JSB $LIBR
      DEF EXIT               DEF TDB
       .               Body of subroutine
       .                      .
       .                      .
EXIT  JMP ENTRY,I      EXIT  JSB $LIBX
      DEF ENTRY              DEF ENTRY
```

Memory-Resident Library

The memory-resident library area in RTE-6/VM contains only Type 6 subroutines that are referenced by memory-resident programs and Type 14 subroutines forced into the memory-resident library at generation time.

Reentrant and privileged subroutines may be placed in the memory-resident library during generation by either of the following methods:

1. If the routine is declared as an external (called) by a memory-resident (Type 1) program, or is called by another memory-resident library subroutine, the subroutine will be automatically placed in the memory-resident library by the generator.

2. The routine can be changed to a Type 14 subroutine during the Parameter Input phase of generation (it also could have been assembled as a Type 14 subroutine).

NOTE

After the relocation of the memory-resident library and all memory-resident programs, all Type 6 routines are converted to Type 7 routines (making them available to disc-resident programs).

Not all subroutines referenced by memory-resident programs are necessarily loaded into the memory-resident library. By declaring the subroutine to be Type 7, the user can ensure that the subroutine will be loaded with the program. Then if .ZRNT and .ZPRV are used instead of $LIBR and $LIBX, the subroutine will execute faster since the system does not need to do the reentrant or privileged processing prior to executing the subroutine.

Utility Subroutine Structure

Utility subroutines are subroutines that cannot be shared by several programs because of internal design or I/O operations. A copy of a utility subroutine is appended to every program that calls it. The PAUSE subroutine is a typical example of a utility subroutine.

When the RTE system is generated, all library subroutines other than Type 8 subroutines are converted to Type 7 utility subroutines following the relocation of memory-resident programs. All required utility subroutines are then relocated immediately following each user program that references them during program relocation.

Format of Routines

The subroutines in each section are presented in the following format:

NOTE

> If a parameter is underlined in a subroutine call description, the value is a variable returned by the system. The parameter value is not supplied by the user.

Name	The name of the routine in the NAM record.
Purpose	The use of the routine.
Entry Points	The entry points to the routine.
Assembly	The Macro/1000 assembly language calling sequence for each entry point. "A" and "B" indicate the A- and B-Registers.
FORTRAN 4X FORTRAN 77	A statement of whether the routine is callable or NOT callable in FORTRAN 4X or FORTRAN 77. Where the routine cannot be called by either version, the statement header is given as FORTRAN with no

version identifiers.

Pascal — A statement of whether the routine is callable or NOT callable in Pascal.

Parameters — An explanation of the parameters form and value.

Result — The type of result and the registers (if any) where it is returned.

Errors — A summary of the error conditions reported by the subroutine. Errors generated by external references are not described. Refer to Appendix A for a fuller discussion of error messages.

Program Type — The three types of library subroutines are Utility (U), Reentrant (R), and Privileged (P).

External References — Other subroutines that are called by the subroutine. All external references except EXEC, $OPSY, REIO, IFBRK, .ZPRV, and .ZRNT are entry points in a library. EXEC and $OPSY are system entry points. IFBRK & REIO are system library entry points.

These symbols receive special handling by the RTE-6/VM generators and loaders. In RTE, both JSB .ZPRV and JSB .ZRNT are changed to RSS unless the routine is generated into the resident library. If the routine is in the resident library, the generator modifies its code as follows:

RQ4CNOP

```
ENTRY NOP           -> ENTRY NOP
      JSB .ZPRV     ->       JSB $LIBR
      DEF EXIT      ->
       :                      :
EXIT  JMP ENTRY,I   -> EXIT  JSB $LIBX
      DEF ENTRY     ->       DEF ENTRY
ENTRY NOP           -> ENTRY NOP
      JSB .ZRNT     ->       JSB $LIBR
      DEF EXIT      ->       DEF TDB
       :                      :
EXIT  JMP ENTRY,I   -> EXIT  JSB $LIBX
      DEF TDB       ->       DEF TDB
      DEC 0         ->       DEC 0
```

$LIBR and $LIBX are system entry points that allow multiple RTE-6/VM programs to share code.

Notes — Additional information for using the routine.

Routines Callable From FORTRAN

Using FORTRAN 4X, routines are callable as a function or subroutine; examples are ABS(x) and RMPAR (IBUF), respectively. The routines must also be in the appropriate calling sequence as well as either the proper function or subroutine format to be callable from FORTRAN 4X; otherwise, they are not callable.

Using FORTRAN 77, some routines are callable under the same conditions as Pascal. Refer to the FORTRAN 77 Reference Manual for a discussion of compatibilities between the two languages.

Routines Callable From Pascal

Using Pascal, routines are callable if the calling sequence is either in the standard calling sequence format or one of the special calling sequences supported by the compiler. For additional information, refer to the Pascal/1000 Reference Manual.

Routines that return results in the A-Register or in the A- and B-Registers are callable from Pascal as functions.

Routines that have names containing characters that are not allowed in Pascal identifiers, such as the leading dot in .RTOI, must have EXTERNAL declarations using the ALIAS compiler option. For example,

```
FUNCTION power
   $ALIAS '.RTOI',DIRECT,ERROREXIT$
   (x=REAL
    i=INT)
   ;REAL;EXTERNAL;
```

All of the routines in this manual operate on one-word addresses only. VAR parameters in HEAP 2 programs should be declared with the $HEAPPARMS OFF$ compiler option in the EXTERNAL declaration.

The calling sequences supported by the compiler are:

1. Standard:
```
   JSB routine
   DEF *+n+1
n(DEF)
```

2. Direct:
```
   JSB routine
n(DEF)
```

3. Standard, Errorexit:
```
   JSB routine
   DEF *+n+1
n(DEF)
   JSB ERRO
```

4. Direct, Errorexit:
```
   JSB routine
n(DEF)
   JSB ERRO
```

A routine is not callable from Pascal if it has any other calling sequence. For example, Pascal does not support passing parameters in registers.

Microcoded Subroutines

Fast FORTRAN Processor

The HP 2100 and 21MX computers have, as an option, a Fast FORTRAN Processor (FFP). The HP 12907 FFP is optional for the HP 2100 computers and the HP 12977 FFP is optional for the HP 21MX computers. The FFP Firmware feature provides for faster execution of the following routines:

.GOTO	..MAP	.ENTR	.ENTP	DBLE
SNGL	.XMPY	.XDIV	.DFER	.XFER
.XADD	.XSUB	.SETP (DOS-III only)		

The following additional Relocatable Subroutine entry points are available only in HP 12977 FFP:

.PWR2	.XPAK	.FLUN	.XCOM	.PACK	..DCM	DDINT

No change from the calling sequence defined in this manual is required to use these routines in FFP, if installed. The user should be aware that after the first execution of a subroutine call, "JSB .GOTO" for example, the main memory location containing the JSB is modified to hold a branch to the ROM address where the .GOTO microcode begins.

Floating Point Library

A second microcode option on the HP 2100 computer, HP 12901 Floating Point, provides firmware for faster execution of the following routines:

.FAD	.FSB	.FMP	.FDV	FLOAT	IFIX

These routines are implemented in the same fashion as the FFP routines. (The HP 21MX computers include the floating point firmware as part of the basic instruction set.)

Chapter 2
Mathematical Subroutines

ABS

Purpose:	Calculate the absolute value of a real x.
Entry Points:	ABS
Assembly:	DLD x JSB ABS Return (result in A & B)
FORTRAN 4X:	Function: ABS (x)
FORTRAN 77:	Function: ABS (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	..FCM, .ZPRV

AIMAG

Purpose:	Extract the imaginary part of a complex x.
Entry Points:	AIMAG
Assembly:	JSB AIMAG DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Function: AIMAG (x)
FORTRAN 77:	Function: AIMAG (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	complex number	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV

AINT

Purpose: Truncate a real x.

Entry Points: AINT

Assembly:
```
DLD x
JSB AINT
Return (result in A & B)
```

FORTRAN 4X: Function: AINT (x)
FORTRAN 77: Function: AINT (x)

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	real to truncate	Real

Result: Real in A & B

Errors: None

Program Type: Type 6, Privileged

External References: .FAD .ZPRV

ALOG

Purpose: Calculate the natural logarithm of a real x.

Entry Points: ALOG

Assembly:
```
DLD x
JSB ALOG
JSB ERRO (error return)
Return (result in A & B)
```

FORTRAN 4X: Function: ALOG (x)
FORTRAN 77: Function: ALOG (x)

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Real in A & B

Errors: x < 0 --> (02 UN)

Program Type: Type 6, Reentrant

External References: .FLUN, FLOAT, .FAD, .FSB, .FDV, .FMP, .ZPRV

ALOGT

Purpose:	Calculate the common logarithm (base 10) of a real x.
Entry Points:	ALOGT ALOGO
Assembly:	DLD x JSB ALOGT (or ALOGO) JSB ERRO (error return) Return (result in A & B)
FORTRAN 4X:	Function: ALOGT (x)
FORTRAN 77:	Function: ALOGT (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B
Errors:	If x <= 0 --> (02 UN)
Program Type:	Type 7, Utility
External References:	ALOG, .FMP

AMOD

Purpose:	Calculate the real remainder of x/y for a real x and y.
Entry Points:	AMOD
Assembly:	JSB AMOD DEF *+3 DEF x DEF y Return (result in A & B)
FORTRAN 4X:	Function: AMOD (x,y)
FORTRAN 77:	Function: AMOD (x,y)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	1st argument	Real
y	2nd argument	Real

Result:	Real in A & B
Errors:	If y = 0, then z = x
Program Type:	Type 6, Privileged
External References:	.ENTP, .ZPRV, AINT, .FDV, .FMP, .FSB

ATAN

Purpose:	Calculate the arctangent of a real x.
Entry Points:	ATAN
Assembly:	DLD x JSB ATAN Return (result in A & B)
FORTRAN 4X:	Function: ATAN (x)
FORTRAN 77:	Function: ATAN (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B (radians)
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ZPRV, ..FCM, .FAD, .FSB, .FDV, .FMP
Notes:	1. Result ranges from -pi/2 to pi/2.

ATAN2

Purpose:	Calculate the real arctangent of the quotient of two reals.
Entry Points:	ATAN2
Assembly:	JSB ATAN2 DEF *+3 DEF y DEF x Return (result in A & B)
FORTRAN 4X:	Function: ATAN2 (y,x)
FORTRAN 77:	Function: ATAN2 (y,x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
y	dividend	Real
x	divisor	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ENTP, SIGN, ATAN, .ZRNT, .FDV, .FAD

CABS

Purpose:	Calculate the real absolute value (modulus) of a complex x.
Entry Points:	CABS
Assembly:	JSB CABS DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Function: CABS (x)
FORTRAN 77:	Function: CABS (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Reentrant
External References:	ABS, .FSB, .FAD, .FDV, .FMP, .ENTP, SQRT, .ZRNT

CEXP

Purpose: Calculate the complex exponential of a complex x.

Entry Points: CEXP

Assembly:
```
JSB CEXP
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: CEXP (x)
FORTRAN 77: Function: CEXP (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	result	Complex

Result: Complex

Errors: None

Program Type: Type 6, Reentrant

External References: .ENTP, EXP, .ZRNT, SIN, COS, .FMP

CLOG

Purpose: Calculate the complex natural logarithm of a complex x.

Entry Points: CLOG

Assembly:
```
JSB CLOG
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: CLOG (x)
FORTRAN 77: Function: CLOG (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	result	Complex

Result: Complex

Errors: If x = 0 --> (02 UN)

Program Type: Type 6, Reentrant

External References: .ENTP, ALOG, .ZRNT, CABS, ATAN2

CMPLX

Purpose: Combine a real x and an imaginary y into a complex z.

Entry Points: CMPLX

Assembly:

```
JSB CMPLX
DEF *+4
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Function: CMPLX (x,y)
FORTRAN 77: Function: CMPLX (x,y)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	real part	Real
y	imaginary part	Real
z	result	Complex

Result: Complex

Errors: None

Program Type: Type 6, Privileged

External References: .ENTP, .ZPRV

CONJG

Purpose: Form the conjugate of a complex x.

Entry Points: CONJG

Assembly:

```
JSB CONJG
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: CONJG (x)
FORTRAN 77: Function: CONJG (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	result	Complex

Result: Complex

Errors: None

Program Type: Type 6, Privileged

External References: .ENTP, ..DLC, .ZPRV

COS

Purpose:	Calculate the sine or cosine of a real x (radians).		
Entry Points:	COS		
Assembly:	DLD x JSB COS Error return Return (result in A & B)		
FORTRAN 4X:	Function:	COS(x)	
FORTRAN 77:	Function:	COS(x)	
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Real
Result:	Real in A & B		
Errors:	x outside [-8192*pi,+8191.75*pi] --> 050R		
Program Type:	Type 6, Reentrant		
External References:	.ZPRV, .CMRS, ..FCM, .FMP, .FAD		

CSNCS

Purpose:	Calculate the complex sine or cosine of a complex x.		
Entry Points:	CSIN, CCOS		
Assembly:	JSB CSIN (or CCOS) DEF *+3 DEF y (result) DEF x JSB error routine Normal return		
FORTRAN 4X:	Function: CSIN (x) or CCOS (x)		
FORTRAN 77:	Function: CSIN (x) or CCOS (x)		
Pascal:	Callable		
Parameters:	Parameter	Description	Type
	x	argument	Complex
	y	result	Complex
Result:	Complex		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	.ENTR, SIN, COS, EXP, ..FCM		

CSQRT

Purpose:	Calculate the complex square root of a complex x.
Entry Points:	CSQRT
Assembly:	JSB CSQRT DEF +*3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: CSQRT (x)
FORTRAN 77:	Function: CSQRT (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	result	Complex

Result:	Complex
Errors:	Overflow bit set if result out of range
Program Type:	Type 6, Reentrant
External References:	.ENTP, ..DLC, .CFER, SQRT, CABS, .ZRNT

DABS

Purpose:	Calculate the absolute value of an extended real x.
Entry Points:	DABS
Assembly:	JSB DABS DEF *+3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: DABS (x)
FORTRAN 77:	Function: DABS (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result:	Extended Real
Errors:	None
Program Type:	Type 6, Reentrant
External References:	..DCM, .DFER, .ENTP. .ZRNT

DATAN

Purpose: Calculate the extended real arctangent of an extended real x.

Entry Points: DATAN

Assembly:

```
JSB DATAN
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DATAN (x)
FORTRAN 77: Function: DATAN (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 6, Reentrant

External References: .ZRNT, .XADD, .XSUB, .XMPY, .XDIV, .ENTP, ..DCM, .FLUN, .DFER

DATN2

Purpose:	Calculate the extended real arctangent of the quotient of two extended reals.
Entry Points:	DATN2 DATA2
Assembly:	JSB DATN2 (or DATA2) DEF *+4 DEF z (result) DEF y DEF x Return
FORTRAN 4X:	Function: DATN2 (y,x)
FORTRAN 77:	Function: DATN2 (y,x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	divisor	Extended Real
y	dividend	Extended Real
z	result	Extended Real

Result:	Extended Real
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ENTP, DSIGN, DATAN, .ZRNT .XADD, .XDIV, .DFER

DBLE

Purpose: Convert a real x to an extended real y.

Entry Points: DBLE

Assembly:

```
JSB DBLE
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DBLE (x)
FORTRAN 77: Function: DBLE (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 6, Privileged

External References: .ZPRV

Notes: This routine is available in firmware. See the description of the Fast FORTRAN Processor (FFP) in Chapter 1.

DCOS

Purpose:	Calculate the extended real cosine of an extended real x (angle in radians).
Entry Points:	DCOS
Assembly:	JSB DCOS DEF *+3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: DCOS (x)
FORTRAN 77:	Function: DCOS (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument in radians	Extended Real
y	result	Extended Real

Result:	Extended Real
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ENTP, DSIN, .ZRNT, .XADD

DDINT

Purpose: Truncate an extended real x to an extended real y.

Entry Points: DDINT

Assembly:
```
JSB DDINT
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DDINT (x)
FORTRAN 77: Function: DDINT (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 6, Reentrant

External References: .XADD, .ENTP, .ZRNT, ENTIX

Notes: This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

DEXP

Purpose: Calculate the extended real exponential of an extended real x.

Entry Points: DEXP

Assembly:

```
JSB DEXP
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: DEXP (x)
FORTRAN 77: Function: DEXP (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: If $e^{x} > (1-2^{-39})\ 2^{127}$ --> (10 OF)

Program Type: Type 6, Reentrant

External References: .ENTP, .XADD, .XSUB, .XMPY, .XDIV, .DFER, .ZRNT, DDINT, SNGL, IFIX, .FLUN, .XPAK

DIM

Purpose: Calculate the positive difference between a real x and y.

Entry Points: DIM

Assembly:
```
JSB DIM
DEF *+3
DEF x
DEF y
Return (result in A & B)
```

FORTRAN 4X: Function: DIM (x,y)
FORTRAN 77: Function: DIM (x,y)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	first argument	Real
y	second argument	Real

Result: Real

Errors: None

Program Type: Type 6, Privileged

External References: .FSB, .ZPRV

DLOG

Purpose:	Calculate the extended real natural logarithm of an extended real x.
Entry Points:	DLOG
Assembly:	JSB DLOG DEF *+3 DEF y (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DLOG (x)
FORTRAN 77:	Function: DLOG (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	first argument	Extended Real
y	second argument	Extended Real

Result:	Extended Real
Errors:	If x <= 0 (11 UN)
Program Type:	Type 6, Reentrant
External References:	.ENTP, .XADD, .XSUB, .XMPY, .XDIV, .FSB, .FLUN, FLOAT, DBLE, .DFER, .ZRNT

DLOGT

Purpose:	Calculate the extended real common logarithm of an extended real x.
Entry Points:	DLOGT (DLOG0)
Assembly:	JSB DLOGT (DLOG0) DEF *+3 DEF y (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DLOGT (x)
FORTRAN 77:	Function: DLOGT (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result:	Extended Real
Errors:	If x < 0 --> (11 UN)
Program Type:	Type 7, Utility
External References:	.ENTP, DLOG, .XMPY

DMOD

Purpose: Calculate the extended real remainder of two extended real values.

Entry Points: DMOD

Assembly:
```
JSB DMOD
DEF *+4
DEF Z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Function: DMOD (x,y)
FORTRAN 77: Function: DMOD (x,y)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	first argument	Extended Real
y	second argument	Extended Real

Result: Extended Real

Errors: If y = 0, then z = x

Program Type: Type 6, Reentrant

External References: .ENTP, .XSUB, .XMPY, .XDIV, DDINT, .ZRNT

DSIGN

Purpose:	Transfer the sign of an extended real y to an extended real x.
Entry Points:	DSIGN
Assembly:	JSB DSIGN DEF *+4 DEF Z (result) DEF x DEF y Return
FORTRAN 4X:	Function: DSIGN (x,y)
FORTRAN 77:	Function: DSIGN (x,y)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	first argument	Extended Real
y	second argument	Extended Real
z	result	Extended Real

Result:	Extended Real
Errors:	If y = 0, z = 0
Program Type:	Type 6, Reentrant
External References:	.DFER, .ENTP, ..DMC, .ZRNT

DSIN

Purpose: Calculate the extended real sine of an extended real x (angle in radians).

Entry Points: DSIN

Assembly:

```
JSB DSIN
DEF *+3
DEF y
DEF x
Return
```

FORTRAN 4X: Function: DSIN (x)
FORTRAN 77: Function: DSIN (x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 6, Reentrant

External References: .ENTP, ..DCM, XPOLY, .DFER, .XSUB., ENTIX, .XADD, .XMPY, .XDIV, .ZRNT

DSQRT

Purpose:	Calculate the extended real square root of an extended real x.
Entry Points:	DSQRT
Assembly:	JSB DSQRT DEF *+3 DEF y (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DSQRT (x)
FORTRAN 77:	Function: DSQRT (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result:	Extended Real
Errors:	If x < 0 --> (03 UN)
Program Type:	Type 6, Reentrant
External References:	.ENTP, DBLE, SNGL, SQRT, .XDIV, .XADD, .ZRNT, .XMPY

DTAN

Purpose:	Calculate tangent of an extended real x.
Entry Points:	DTAN
Assembly:	JSB DTAN DEF *+3 DEF y (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DTAN (x)
FORTRAN 77:	Function: DTAN (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real (radians)
y	result	Extended Real

Result:	Extended Real
Errors:	X outside [-8192*pi,+8191.75*pi] --> 09 OR
Program Type:	Type 7, Utility
External References:	.ENTR, .DFER, .TMPY, .TSUB, .TINT, .ITBL, .XADD, .XMPY, .XDIV, XPOLY

DTANH

Purpose: Calculate hyperbolic tangent of an extended real x.

Entry Points: DTANH

Assembly:
```
JSB DTANH
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DTANH (y,x)
FORTRAN 77: Function: DTANH (y,x)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 7, Utility

External References: .ENTR, .DFER, .XFER, .FLUN, .PWRZ, DEXP, .XADD, .XMPY, .XDIV

ENTIE

Purpose:

1. Calculate the greatest integer not algebraically exceeding a real x (ENTIE);
2. Round a real x to the nearest integer; if half way between two integers, select the algebraically larger integer (.RND).

Entry Points: ENTIE, .RND

Assembly:

```
DLD x
JSB .RND (or ENTIE)
Return (Result in A)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Two Integers: sign in A; Integer in B

Errors: See Notes.

Program Type: Type 7, Utility

External References: None

Notes:

```
If exponent  >= 0 then A = 32767
    else               A = 32768
    Result: Integer in A
```

ENTIX

Purpose:	Calculate ENTIER of an extended real x.
Entry Points:	.XENT, ENTIX
Assembly:	JSB .XENT(or ENTIX) DEF *+3 DEF y (result) DEF x Return
FORTRAN:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Extended Real

Result:	Extended Real
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ENTP, .ZPRV

EXP

Purpose: Calculate e^x, where x is real.

Entry Points: EXP

Assembly:
```
DLD x
JSB EXP
JSB ERR0 (error)
Return (result in A & B)
```

FORTRAN 4X: Function: EXP (x)
FORTRAN 77: Function: EXP (x)

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Real in A & B

Errors: $x*\log_2 e$ >= 127 --> (07 OF)

Program Type: Type 6, Reentrant

External References: .ZPRV, .CMRS, .PWRZ, .FMP, .FSB, .FAD, .FDV

FADSB

Purpose:	.FAD: Add real x to y. .FSB: Subtract real y from x.
Entry Points:	.FAD, .FSB
Assembly:	DLD x JSB .FAD Return (Result in A & B) or DLD x JSB .FSB Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable
Parameters:	See table below
Result:	Real
Errors:	See Notes.
Program Type:	Type 6, Privileged
External References:	.PACK, .ZPRV

Parameter	Description	Type
x	argument	Real

FLOAT

Purpose:	Convert integer i to a real x.		
Entry Points:	FLOAT		
Assembly:	LDA i JSB FLOAT Return (result in A & B)		
FORTRAN 4X:	Function: FLOAT (i)		
FORTRAN 77:	Function: FLOAT (i)		
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	i	argument	Integer
Result:	Real in A & B		
Errors:	None		
Program Type:	Type 6, Privileged		
External References:	.PACK, .ZPRV		

IABS

Purpose:	Calculate absolute value of integer i.
Entry Points:	IABS
Assembly:	LDA i JSB IABS Return (Result in A)
FORTRAN 4X:	Function: IABS (i)
FORTRAN 77:	Function: IABS (i)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
i	argument	Integer

Result:	Integer in A
Errors:	See Notes.
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	If IABS is (-32768), the result is 32767 and the overflow bit is set.

IAND

Purpose:	Take the logical product and integers i and j.
Entry Points:	IAND
Assembly:	JSB IAND DEF i DEF j Return (Result in A)
FORTRAN 4X:	Function: IAND (i,j)
FORTRAN 77:	Function: IAND (i,j)
Pascal:	Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	argument	Integer

Result:	Integer in A
Errors:	None
Program Type:	Type 7, Utility
External References:	None

IDIM

Purpose:	Calculate the positive difference between integers i and j.
Entry Points:	IDIM

Assembly:

```
JSB IDIM
DEF *+3
DEF i
DEF j
Return (Result in A)
```

FORTRAN 4X: Function: IDIM (i,j)
FORTRAN 77: Function: IDIM (i,j)

Pascal: Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	argument	Integer

Result: Integer in A

Errors: See Notes.

Program Type: Type 6, Privileged

External References: .ZPRV

Notes: If IDIM(i,j) is out of range, the overflow bit is set and a value of 32767 returned.

IDINT

Purpose:	Truncate an extended real to an integer.
Entry Points:	IDINT
Assembly:	JSB IDINT DEF *+2 DEF x Return (Result in A)
FORTRAN 4X:	Function: IDINT (x)
FORTRAN 77:	Function: IDINT (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real

Result:	Integer in A
Errors:	If IDINT (x) is out of range, then result = 32767 and the overflow bit is set.
Program Type:	Type 6, Privileged
External References:	IFIX, .ZPRV, SNGM

IFIX

Purpose: Convert a real x to an integer.

Entry Points: IFIX (P)

Assembly:

```
DLD x
JSB IFIX
Return (result in A)
```

FORTRAN 4X: Function: IFIX (x)
FORTRAN 77: Function: IFIX (x)

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Integer in A (see notes)

Errors: None

Program Type: Type 6, Privileged

External References: Non-floating point library: .FLUN

Floating point library: .ZPRV

Notes:

1. Any fractional portion of the result is truncated. If the integer portion is greater than or equal to 2^{15}, the result is set to 32767.
2. The routine IFIX exists only in non-floating point libraries.

INT

Purpose:	Truncate a real x to an integer.
Entry Points:	INT
Assembly:	DLD x JSB INT Return (Result in A)
FORTRAN 4X:	Function: INT (x)
FORTRAN 77:	Function: INT (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Integer in A
Errors:	If INT (x) is out of range, the overflow bit is set. The result is set to 32767.
Program Type:	Type 7, Utility
External References:	IFIX

IOR

Purpose:	Take logical inclusive OR of integers i and j.		
Entry Points:	IOR		
Assembly:	JSB IOR DEF i DEF j Return (Result in A)		
FORTRAN 4X:	Function: IOR (i,j)		
FORTRAN 77:	Function: IOR (i,j)		
Pascal:	Callable		
Parameters:	Parameter	Description	Type
	i	argument	Integer
	j	argument	Integer
Result:	Integer in A		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	None		

ISIGN

Purpose:	Calculate the sign of z times the absolute value of i, where z is real or integer and i is integer.
Entry Points:	ISIGN
Assembly:	JSB ISIGN DEF i DEF z Return (Result in A)
FORTRAN 4X:	Function: ISIGN (i,z)
FORTRAN 77:	Function: ISIGN (i,z)
Pascal:	Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
z	argument	Real

Result:	Integer in A
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV

IXOR

Purpose: Perform an integer exclusive OR.

Entry Points: IXOR

Assembly:

```
JSB IXOR
DEF *+3
DEF i
DEF j
Return (Result in A)
```

FORTRAN 4X: Function: IXOR (i,j)
FORTRAN 77: Function: IXOR (i,j)

Pascal: Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	argument	Integer

Result: Integer in A

Errors: None

Program Type: Type 7, Utility

External References: None

MOD

Purpose:	Calculate the integer remainder of i/j for integer i and j.
Entry Points:	MOD
Assembly:	JSB MOD DEF *+3 DEF i DEF j Return (result in A & B)
FORTRAN 4X:	Function: MOD (i,j)
FORTRAN 77:	Function: MOD (i,j)
Pascal:	Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	argument	Integer

Result:	Integer in A
Errors:	If j=0, then result = i
Program Type:	Type 6, Privileged
External References:	.ZPRV

MXMND

Purpose: Calculate the maximum or minimum of a series of extended real values.

Entry Points: DMAX1, DMIN1

Assembly:

```
JSB DMAX1 (or DMIN1)
DEF *+n+2
DEF y (result)
DEF a
DEF b
 :  :
DEF n
Return
```

FORTRAN: See notes.

Pascal: Callable

Parameters:

Parameter	Description	Type
a	argument	Extended Real
b	argument	Extended Real
:	:	:
n	argument	Extended Real
y	result	Extended Real

Result: Extended Real

Errors: If $n < 2$, then $y = 0$.

Program Type: Type 7, Reentrant

External References: .XSUB, .DFER

Notes: Versions FORTRAN 4X and FORTRAN 77 intrinsic functions:

```
DMAX1 (a,b,c,....)
DMIN1 (a,b,c,....)
```

MXMNI

Purpose: Calculate the maximum or minimum of a series of integer values.

Entry Points: AMAX0, MAX0, AMIN0, MIN0

Assembly:
```
JSB entry point
DEF *+n+1
DEF a
DEF b
 :  :
DEF n
Return (Result in A or A & B)
```

FORTRAN: See notes.

Pascal: Callable

Parameters:

Parameter	Description	Type
a	argument	Integer
b	argument	Integer
:	:	:
n	argument	Integer

Result: Real in A & B for AMAX0 and AMIN0

Integer in A for MAX0 and MIN0

Errors: If the number of parameters is less than 2,

y = 0

Program Type: Type 7, Utility

External References: FLOAT

Notes: Versions FORTRAN 4X and FORTRAN 77 functions:

```
AMAX0 (a,b,...,n),
MAX0 (a,b,...,n),
AMIN0 (a,b,...,n),
MIN0 (a,b,...,n).
```

MXMNR

Purpose: Calculate the maximum or minimum of a series of real values.

Entry Points: AMAX1, MAX1, AMIN1, MIN1

Assembly:

```
JSB Entry Point
DEF *+ n+1
DEF a
DEF b
 :  :
DEF n
Return (y in A or A & B)
```

FORTRAN: See notes.

Pascal: Callable

Parameters:

Parameter	Description	Type
a	argument	Real
b	argument	Real
:	:	:
n	argument	Real

Result: Real in A & B for AMAX1 and AMIN1

Integer in A for MAX1 and MIN1

Errors: If the number of parameters is less than 2, y = 0.

Program Type: Type 7, Utility

External References: IFIX, .FSB

Notes:

1. Callable as integer or real procedure, but only with a fixed number of parameters.

2. FORTRAN 4X and FORTRAN 77 functions:

   AMAX1(a,b,...,n),MAX1(a,b,...,n),
   AMIN(a,b,...,n),MIN1(a,b,...,n).

REAL

Purpose:	Extract the real part of a complex x.
Entry Points:	REAL
Assembly:	JSB REAL DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Function: REAL (x)
FORTRAN 77:	Function: REAL (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV

SIGN

Purpose:	Calculate the sign of z times the absolute value of x, where z is real or integer and x is real.
Entry Points:	SIGN
Assembly:	JSB SIGN DEF x DEF z Return (result in A & B)
FORTRAN 4X:	Function: SIGN (x,z)
FORTRAN 77:	Function: SIGN (x,z)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Real
z	argument	Integer or Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	..FCM, .ZPRV

SIN

.SNCS

Purpose:	Calculate the sine of a real x (radians).		
Entry Points:	SIN		
Assembly:	DLD x JSB SIN Error return Return (result in A & B)		
FORTRAN 4X:	Function:	SIN(x)	
FORTRAN 77:	Function:	SIN(x)	
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Real
Result:	Real in A & B		
Errors:	x outside [-8192*pi,+8191.75*pi] --> 05OR		
Program Type:	Type 6, Reentrant		
External References:	.ZPRV, .CMRS, ..FCM, .FMP, .FAD		

SNGL

Purpose:	Convert an extended real x to a real y.
Entry Points:	SNGL
Assembly:	JSB SNGL DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Function: SNGL (x)
FORTRAN 77:	Function: SNGL (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	The routine is available in firmware. See the description of the Fast FORTRAN Processor (FFP) in Chapter 1.

SNGM

Purpose:	Convert an extended real x to a real y without rounding.
Entry Points:	SNGM
Assembly:	JSB SNGM DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Function: SNGM (x)
FORTRAN 77:	Function: SNGM (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real

Result:	Real in A & B
Errors:	If $y < ABS((-1+2^{-23})*2^{-128})$, zero is returned.
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	Maximum error will be less than the least significant bit.

SQRT

Purpose:	Calculate the square root of a real x.
Entry Points:	SQRT
Assembly:	DLD x JSB SQRT JSB ERR0 (error) Return (result in A & B)
FORTRAN 4X:	Function: SQRT (x)
FORTRAN 77:	Function: SQRT (x)
Pascal:	Not callable
Parameters:	Parameter / Description / Type: x / argument / Real
Result:	Real in A & B
Errors:	x < 0 -> (03 UN)
Program Type:	Type 6, Reentrant
External References:	.ZPRV, .FLUN, .PWR2, .FMP, .FAD, .FDV

TAN

Purpose:	Calculate the tangent of a real x (radians).
Entry Points:	TAN
Assembly:	DLD x JSB TAN JSB ERR0 (error) Return (result in A & B)
FORTRAN 4X:	Function: TAN (x)
FORTRAN 77:	Function: TAN (x)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument (radians)	Real

Result:	Real in A & B
Errors:	x outside [-8192*pi,+8191.75*pi] --> 09 OR
Program Type:	Type 6, Reentrant
External References:	.ZPRV, .CMRS, .FMP, .FAD, .FDV

TANH

Purpose:	Calculate the hyperbolic tangent of a real x.
Entry Points:	TANH
Assembly:	DLD x JSB TANH Return (result in A & B)
FORTRAN 4X:	Function: TANH (x)
FORTRAN 77:	Function: TANH (x)
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ZPRV, .EXP, .FAD, .FSB, .FDV, .FMP

DPOLY

Purpose: Evaluate the quotient of two polynomials in double precision.

Entry Points: DPOLY, TRNL

Assembly:

```
JSB DPOLY                  OR JSB DPOLY
DEF *+6                       OCT <flags>
DEF z (result)                DEF z (result)
DEF x                         DEF x
DEF c                         DEF c
DEF m                         DMF m
DEF n                         DEF n
Return                        Return
```

FORTRAN 4X: CALL DPOLY (z,x,c,m,n)
FORTRAN 77: CALL DPOLY (z,x,c,m,n)

Pascal: Callable

Parameters:

Parameter	Description	Type
z	result	Double Real
x	argument	Double Real
c	coefficient list	Address
m	order of numerator	Integer
n	order of denominator	Integer

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .ENTR, .CFER, .TADD, .TSUB, .TMPY, .TDIV, .4XRO

XADSB

Purpose:	Extend the real addition and subtraction.
Entry Points:	.XADD, .XSUB
Assembly:	JSB (.XADD or .XSUB) DEF z (result) DEF x DEF y Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	argument	Extended Real
z	result	Extended Real

Result:	Extended Real
Errors:	See notes.
Program Type:	Type 6, Privileged
External References:	.XPAK, ADRES, .ZPRV
Notes:	These routines are available in firmware. See the description of the Fast FORTRAN Processor (FFP) in Chapter 1.

XPOLY

Purpose: Evaluate the extended real polynomial.

Entry Points: .XPLY, XPOLY

Assembly:
JSB .XPLY or XPOLY
DEF *+5
DEF y (result)
DEF n (degree + 1)
DEF x
DEF c_1 (first element of coefficient array)
Return

FORTRAN 4X: .XPLY - Not callable
XPOLY - Callable

FORTRAN 77: Callable

Pascal: Callable

Result: Extended Real

Errors: If n <= 0, y = 0

Program Type: Type 6, Reentrant

External References: .ZRNT, .ENTP, .XADD, .XMPY, .DFER

Notes: These descriptions are available in firmware. See the description of the Fast FORTRAN Processor (FFP) in Chapter 1.

.ABS

Purpose: Find the absolute value of a double real.

Entry Points: .ABS

Assembly:

```
JSB ABS
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DABS (with Y option)
FORTRAN 77: Function: DABS (with Y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	argument	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .CFER, .TSUB, .4ZRO, .ENTR

.ATAN

Purpose:	Calculate the inverse tangent of a double real x.
Entry Points:	.ATAN
Assembly:	JSB .ATAN DEF *+3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: DATAN (with Y option)
FORTRAN 77:	Function: DATAN (with Y option)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result:	Double Real (radians)
Errors:	None
Program Type:	Type 7, Utility
External References:	.ENTR, CRER, TRNC, .TDIV, ..TCM, .FLUN, .TSUB, /ATCG

.ATN2

Purpose: Calculate the arctangent of the quotient x/y of two double real variables x and y.

Entry Points: .ATN2, .ATA2

Assembly:
```
JSB .ATN2
DEF *+4
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Function: DATN2 or DATAN2 (with Y option)
FORTRAN 77: Function: DATN2 or DATAN2 (with Y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	argument	Double Real
z	result	Double Real

Result: Double Real (radians)

Errors: x = y = 0 gives error code 15 UN

Program Type: Type 7, Utility

External References: .ATAN, .TADD, .TSUB, .TDIV, .ENTR, .4ZERO, .CFER

.BLE

Purpose: Convert real x to double real y.

Entry Points: .BLE

Assembly:

```
JSB .BLE
DEF *+3
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Function: DBLE (with Y option)
FORTRAN 77: Function: DBLE (with Y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	result	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .ENTR

.CADD

Purpose: Add complex x to complex y.

Entry Points: .CADD

Assembly:
```
JSB .CADD
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	argument	Complex
z	result	Complex

Result: Complex

Errors: Overflow bit set if result out of range.

Program Type: Type 6, Reentrant

External References: .ETNC, .ZRNT, .FAD

Notes: If (OVF(IDMY)) 10,20

10 - start of user overflow-set routine
20 - start of user overflow-clear routine

(See OVF Function.)

.CDBL

Purpose: Extract the real part of a complex x and return it as an extended precision real y.

Entry Points: .CDBL

Assembly:
```
JSB .CDBL
DEF y
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 7, Utility

External References: DBLE

.CDIV

Purpose: Divide complex x by complex y.

Entry Points: .CDIV

Assembly:
```
JSB .CDIV
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	argument	Complex
z	result	Complex

Result: Complex

Errors: Overflow bit set if result out of range.

Program Type: Type 6, Reentrant

External References: .ZRNT, .ENTC

.CFER

Purpose:	Move four words from address x to address y. Used to tranfer a complex x to complex y.
Entry Points:	.CFER
Assembly:	JSB .CFER DEF y DEF x Return A = direct address of (x+4) B = direct address of (y+4) Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	source	Complex
y	destination	Complex

Result:	Complex
Errors:	None
Program Type:	Type 6, Utility
External References:	.ZPRV

.CHEB

Purpose:	Evaluate the Chebyshev series at a real x for a particular table of coefficients c.
Entry Points:	.CHEB
Assembly:	DLD x JSB .CHEB DEF c (table, note 1) Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable
Result:	Real
Errors:	None
Program Type:	Type 6, Reentrant
External References:	.ZRNT, .FAD, .FMP, .FSB
Notes:	Table c consists of a series of real coefficients terminated by an integer zero.

.CINT

Purpose:	Convert the real part of a complex x to an integer.
Entry Points:	.CINT
Assembly:	JSB .CINT DEF x Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex

Result:	Integer in A
Errors:	None
Program Type:	Type 7, Utility
External References:	IFIX

.CMPY

Purpose: Multiply a complex x by a complex y.

Entry Points: .CMPY

Assembly:
```
JSB .CMPY
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	argument	Complex
z	result	Complex

Result: Complex

Errors: Overflow bit set if result out of range.

Program Type: Type 6, Reentrant

External References: .ZRNT, .ENTC

.CMRS

Purpose: Reduce the argument for SIN, COS, TAN, EXP

Entry Points: .CMRS

Assembly:

```
DLD x
JSB .CMRS
DEF
DEF N
Error return
Normal return  (Real result in A & B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real
c	argument	Extended Real
n	result	Integer

Result: Real and Extended Precision

Errors: N outside $[-2^{15}, 2^{15})$ gives error return

Program Type: Type 6, Reentrant

External References: .ZPRV, .XMPY, .XSUB, SNGL, IFIX, FLOAT

.CSUB

Purpose:	Subtract a complex y from a complex x.
Entry Points:	.CSUB

Assembly:

```
JSB .CSUB
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
y	argument	Complex
z	result	Complex

Result:	Complex
Errors:	Overflow bit set if result out of range
Program Type:	Type 6, Reentrant
External References:	.ENTC, .ZRNT

.CTBL

Purpose:	Convert a complex real to a double real.		
Entry Points:	.CTBL		
Assembly:	JSB .CTBL DEF y (result) DEF x Return		
FORTRAN 4X:	Not callable		
FORTRAN 77:	Callable		
Pascal:	Callable		
Parameters:	Parameter	Description	Type
	x	argument	Complex
	y	result	Double Real
Result:	Double Real		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	.BLE		

.CTOI

Purpose:	Raise a complex x to an integer power i.
Entry Points:	.CTOI
Assembly:	JSB .CTOI DEF z (result) DEF x DEF i Error return Normal return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex
i	exponent	Integer
z	result	Complex

Result:	Complex
Errors:	x = 0, i<=0 --> (14 UN)
Program Type:	Type 6, Reentrant
External References:	.CMPY, .CDIV, .CFER, .ENTC, .ZRNT

.DCPX

Purpose: Convert an extended real x to a complex y.

Entry Points: .DCPX

Assembly:

```
JSB .DCPX
DEF Y
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	result	Complex

Result: Complex

Errors: None

Program Type: Type 7, Utility

External References: SNGL, CMPLX

.DFER

Purpose:	Extend a real transfer.		
Entry Points:	.DFER		
Assembly:	JSB .DFER DEF y DEF x Return A = direct address of x+3 B = direct address of y+3		
FORTRAN 4X:	Not callable		
FORTRAN 77:	Callable		
Pascal:	Callable		
Parameters:	Parameter	Description	Type
	x	source	Extended Real
	y	destination	Extended Real
Result:	Extended Real		
Errors:	None		
Program Type:	Type 6, Privileged		
External References:	.ZPRV		
Notes:	This routine is available in firmware. See the description of the Fast FORTRAN Processor (FFP) in Chapter 1. .DFER (2100 MICROCODE) returns x+4, y+4 in A,B-Registers. .DFER (21MX MICROCODE) returns x+3, y+3 in A,B-Registers.		

.DINT

Purpose:	Convert a double real x to an integer.
Entry Points:	.DINT, .XFTS
Assembly:	JSB .DINT DEF x Return (Result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result:	Integer in A
Errors:	None
Program Type:	Type 6, Utility
External References:	SNGM, IFIX, .ZPRV

.DIV

Purpose: Replace the subroutine call with the hardware instruction to divide a double integer i by the integer j.

Entry Points: .DIV

Assembly:

```
DLD i
JSB DIV
DEF j
Return (result in A, remainder in B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
i	dividend	Double Integer
j	divisor	Integer

Result: Integer quotient in A and remainder in B

Errors: -32768>quotient>32767 --> overflow, quotient <--32767

Program Type: Type 7, Utility

External References: .MAC.

Notes:

1. The DLD loads the value i into the A- and B-Registers with the sign and 15 most significant bits in B and the least significant bits in A.

2. Since the subroutine call is replaced by the hardware instructions, the routine is entered only once for each subroutine call.

.DLD

Purpose:	Replace the subroutine call with the hardware instruction to load the contents of memory locations x and x+1 into the A- and B-Registers, respectively.
Entry Points:	.DLD
Assembly:	JSB .DLD or DLD x DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Parameters:	

Parameter	Description	Type
x	memory location	

Result:	Two-word quantity: A & B
Errors:	None
Program Type:	Type 7, Utility
External References:	.MAC.
Notes:	Since the subroutine call is replaced by the hardware instruction, the routine is entered only once for each subroutine call.

.DST

Purpose:	Replace the subroutine call with the hardware instruction to store the contents of the A- and B-Registers in memory locations x and x+i.
Entry Points:	.DST
Assembly:	JSB .DST DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	memory location	

Result:	Two-word quantity
Errors:	None
Program Type:	Type 7, Utility
External References:	.MAC.
Notes:	Since the subroutine call is replaced by the hardware instruction, the routine is entered only once for each subroutine call.

.DTOD

Purpose: Raise a double real x to a double real power y.

Entry Points: .DTOD

Assembly:

```
JSB .DTOD
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	exponent	Double Real
z	result	Double Real

Result: Double Real

Errors: See Notes

Program Type: Type 6, Reentrant

External References: DEXP, DLOG, .XMPY, .DFER, .ENTC, .ZRNT

Notes:

$x = 0,\ y <= 0 \quad --> \quad (13\ UN)$

$x < 0,\ Y <= 0 \quad --> \quad (13\ UN)$

$x > (1-2^{-39})2^{127} \quad --> \quad (10\ OF)$

.DTOI

Purpose:	Calculate an extended real x raised to an integer power i.
Entry Points:	.DTOI
Assembly:	JSB .DTOI DEF y (result) DEF x DEF I Error return Normal return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
i	exponent	Integer
y	result	Extended Real

Result:	Extended Real
Errors:	If x = 0, I <= 0 --> (12 UN)
Program Type:	Type 6, Reentrant
External References:	.XMPY, .XDIV, .DFER, .ZRNT

.DTOR

Purpose: Raise a double real x to a real power y.

Entry Points: .DTOR

Assembly:
```
JSB .DTOR
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	exponent	Real
z	result	Double Real

Result: Double Real

Errors: If x = 0, I <= 0 --> (12 UN)

Program Type: Type 7, Utility

External References: .DTOD, DBLE

.EXP

Purpose:	Calculate e^x where x is double real
Entry Points:	.EXP
Assembly:	JSB .EXP DEF *+3 DEF z (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DEXP (x) (with Y option)
FORTRAN 77:	Function: DEXP (x) (with Y option)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	exponent	Double Real
y	result	Double Real

Result:	Double Real
Errors:	x > 127*LN(2) gives error code 07 OF
Program Type:	Type 7, Utility
External References:	.ENTR, .CFER, .4ZRO, /CMRT, /EXTH
Notes:	For x < -129*LN(2), a zero will be returned with no error indication.

.FDV

Purpose:	Divide a real x by y.
Entry Points:	.FDV
Assembly:	DLD x JSB .FDV DEF y Return (quotient in A & B, O set if under/overflow)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	dividend	Real
y	divisor	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.PACK, .ZPRV

.FLUN

Purpose: "Unpack" a real x; place exponent in A, lower part of mantissa in B.

Entry Points: .FLUN

Assembly:
```
DLD x
JSB .FLUN
Return exponent in A
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Lower mantissa in B

Errors: None

Program Type: Type 6, Privileged

External References: .ZPRV

Notes: This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

.FMP

Purpose:	Multiply a real x by y.
Entry Points:	.FMP
Assembly:	DLD y JSB .FMP DEF x Return (product in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	argument	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.PACK, .ZPRV

.FPWR

Purpose:	Calculate x^i for real x and unsigned integer i.
Entry Points:	.FPWR
Assembly:	LDA i JSB .FPWR DEF x Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real
i	exponent	Unsigned Integer

Result:	Real in A & B
Errors:	None
Program Type:	Type 7, Utility
External References:	.FMP, FLOAT, .FLUN

Notes:

1. "i" must be in the range [2,32768].
2. If overflow occurs, the maximum positive number is returned with overflow set. Overflow is set if underflow occurs.
3. The X- and Y-Registers may be altered.

.ICPX

Purpose: Convert an integer i to a complex y.

Entry Points: .ICPX

Assembly:

```
LDA i
JSB .ICPX
DEF y
Return
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
i	argument	Integer
y	result	Complex

Result: Complex

Errors: None

Program Type: Type 7, Utility

External References: FLOAT, CMPLX

.IDBL

Purpose: Convert an integer i to an extended real y.

Entry Points: .IDBL, .XFTS

Assembly:

```
LDA i
JSB .IDBL
DEF y
Return
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
i	argument	Integer
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 7, Utility

External References: FLOAT, DBLE

.IENT

Purpose:	Calculate the greatest integer not algebraically exceeding a real x.
Entry Points:	.IENT
Assembly:	DLD x JSB .IENT JSB error routine Return (result in A)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Integer in A
Errors:	EXPO (x) > 14, user must supply error routine.
Program Type:	Type 6, Privileged
External References:	IFIX, .FLUN, FLOAT, .ZPRV

.ITBL

Purpose:	Convert an integer x to a double real y.
Entry Points:	.ITBL, .TFTS
Assembly:	LDA x JSB .ITBL DEF y (result) Return
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Integer
y	result	Double Real

Result:	Double Real
Errors:	None
Program Type:	Type 7, Utility
External References:	.BLE, FLOAT

.ITOI

Purpose: Calculate i^j for integer i and j: $k = i^j$.

Entry Points: .ITOI

Assembly:
```
JSB .ITOI
DEF i
DEF j
JSB ERR0 (error return)
Return (result in A)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	exponent	Integer

Result: Integer in A

Errors: None

Program Type: Type 6, Privileged

External References: .ZPRV

.LBT

Purpose:	Load A with the lower byte at the word contained at the address of B.
Entry Points:	.LBT
Assembly:	LDB x JSB .LBT Return
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	

Result:	A
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	A byte address is defined as two times the word address of the memory location containing the byte of data. If the byte is in bits 0 to 7, bit 0 of the byte address is set; if the byte is in bits 8 - 15, bit 0 of the byte address is clear.

.LOG

Purpose:	Calculate the natural logarithm of a double real x.
Entry Points:	.LOG
Assembly:	JSB .LOG DEF *+3 DEF y (result) DEF x Error return Normal return
FORTRAN 4X:	Function: DLOG (with Y option)
FORTRAN 77:	Function: DLOG (with Y option)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result:	Double Real
Errors:	X < 0 --> 02 UN
Program Type:	Type 7, Utility
External References:	.ENTR, .CFER, .FLUN, .TADD, .TMPY, TRNL, /ATLG, FLOAT

.LOGO

Purpose:	Calculate the common (base 10) logarithm of a double real x.
Entry Points:	.LOGO (.LOGT)

Assembly:

```
JSB .LOGO (.LOGT)
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: DLOGT (or DLOG10) (with Y option)
FORTRAN 77: Function: DLOGT (or DLOG10) (with Y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors: x <= 0 gives error code 02 UN

Program Type: Type 7, Utility

External References: .LOG, .TMPY, .ENTR

.MAC.

Purpose: Replace a JSB .subr with a machine language Macro jump 105nnn that initiates firmware.

Entry Points: .MAC.

Assembly:

```
.subr NOP
      JSB .MAC.
      OCT 105nnn
      END
```

where nnn is between 000 and 377

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: In-line code change

Errors: None

Program Type: Type 7, Utility

External References: None

Notes: The same result is achieved in RTE during system generation using the Replace Command.

.MANT

Purpose:	Extract the mantissa of a real x.
Entry Points:	.MANT
Assembly:	DLD x JSB .MANT Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable
Parameters:	Parameter / Description / Type: x / argument / Real
Result:	Real mantissa in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV

.MOD

Purpose:	Calculate the remainder of x/y, where x,y and result are double reals.
Entry Points:	.MOD

Assembly:

```
JSB .MOD
DEF *+4
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Function: DMOD (x,y) (with y option)
FORTRAN 77: Function: DMOD (x,y) (with y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	dividend	Double Real
y	divisor	Double Real
z	result	Double Real

Result: Double Real

Errors: If y = 0 then the result is zero.

Program Type: Type 7, Utility

External References: .CFER, .TSUB, .TMPY, .TDIV, .YINT, .ENTR, .4ZRO

Notes:

1. The function .MOD will return x if y=0, or x/y overflows or underflows.
2. If an overflow or underflow occurs elsewhere in the calculation, the result will be incorrect.
3. No attempt is made to recover precision lost in the subtract.

.MPY

Purpose:	Replace the subroutine call with the hardware instruction to multiply integer i and j.
Entry Points:	.MPY
Assembly:	LDA j JSB .MPY DEF i Return (result in A & B) (see notes)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
i	argument	Integer
j	argument	Integer

Result:	Double Integer in A & B
Errors:	None
Program Type:	Type 7, Utility
External References:	.MAC.

Notes:

1. B contains most significant bits of product; A contains least significant bits.
2. Since the subroutine call is replaced by the hardware instruction, the routine is called only once for each subroutine call.

.MXMN

Purpose: Find the maximum (or minimum) of a list of double reals.

Entry Points: .MAX1, .MIM1

Assembly:

```
JSB .MAX1          OR  JSB .MIN1
DEF *+N+2              DEF *+N+2
DEF                    DEF
DEF A                  DEF A
DEF B                  DEF B
 :  :                   :  :
DEF N                  DEF N
Return                 Return
```

FORTRAN 4X: Functions: DMAX1 (with y option)
DMIN1 (with y option)

FORTRAN 77: Functions: DMAX1 (with y option)
DMIN1 (with y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
A,B,...,N	argument	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .CFER, .TSUB, .4ZRO

Notes:

1. If there is only one argument in the list, it is considered to be both the maximum and minimum of the list.

2. If the list is null, zero will be returned.

.NGL

Purpose:	Convert double real x to real.
Entry Points:	.NGL
Assembly:	JSB .NGL DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 7, Utility
External References:	SNGL, .CFER
Notes:	The result is rounded unless this would cause overflow. If so, overflow is set and the result is truncated to the greatest positive number.

.PACK

Purpose:	Convert the signed mantissa of a real x into normalized real format.
Entry Points:	.PACK
Assembly:	DLD x JSB .PACK BSS 1 (exponent) Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real
BSS 1	exponent returned	Integer

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	This routine is available in 21MX FMP firmware. See Chapter 1.

.PWR2

Purpose:	Calculate for real x and integer n.
Entry Points:	.PWR2
Assembly:	DLD x JSB .PWR2 DEF n Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real
n	exponent	Integer

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

.RTOD

Purpose: Raise a real x to a double real power y.

Entry Points: .RTOD

Assembly:

```
JSB .RTOD
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	exponent	Double Real
z	result	Double Real

Result: Double Real

Errors: See Notes.

Program Type: Type 7, Utility

External References: .DTOD, DBLE

Notes:

x = 0, y <=0 --> (13 UN)
x < 0, Y <= 0 --> (13 UN)
$x > (1-2^{-39})2^{127}$ --> (10 OF)

.RTOI

Purpose: Calculate x^i for real x and integer i.

Entry Points: .RTOI

Assembly:

```
JSB .RTOI
DEF x
DEF i
JSB ERR0
Return (result in A & B)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
i	exponent	Integer

Result: Real in A & B

Errors:

Condition	Error code
x = 0, i <= 0	06 UN
$x^{\lvert i \rvert} > 2^{127}$	(floating point overflow)

Program Type: Type 7, Utility

External References: .FPWR, .FDV

.RTOR

Purpose: Calculate x^y for a real x and y.

Entry Points: .RTOR

Assembly:
```
JSB .RTOR
DEF x
DEF y
JSB ERR0
Return (result in A & B)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	exponent	Real

Result: Real in A & B

Errors:

Condition	Error Code
x = 0, y <= 0 < 0, = 0	04 UN
|x*ALOG(x) | >= 124	07 OF

On error return, the overflow bit is set.

Program Type: Type 6, Reentrant

External References: ALOG, EXP, .ZRNT, .FMP

.RTOR

Purpose: Calculate x^y, where x is a real and y is a double real.

Entry Points: .RTOT

Assembly:
```
JSB .RTOT
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Real
y	exponent	Double Real
z	result	Double Real

Result: Double Real

Errors: See Notes.

Program Type: Type 7, Utility

External References: .TTOT

Notes: Underflow will give a zero result, with no error. Overflow returns the greatest positive number, sets overflow (cleared otherwise), and gives an error code of 07 OF.

If (x<0) or (x=0 and Y<=0) there will be an error code of 13 UN.

.SBT

Purpose: Replace a 21MX microcoded instruction SBT. Store the lower byte of the A into the address contained in B.

Entry Points: .SBT

Assembly:

```
LDB x
JSB .SBT
Return
```

FORTRAN: Not callable

Pascal: Not callable

Result: A

Errors: None

Program Type: Type 6, Privileged

External References: .ZPRV

Notes: A byte address is defined as two times the word address of the memory location containing the byte of data. If the byte is in bits 0 to 7, bit 0 of the byte address is set; if the byte is in bits 8 - 15, bit 0 of the byte address is clear.

.SIGN

Purpose: Transfer the sign of a double real y to a double real x.

Entry Points: .SIGN

Assembly:
```
JSB .SIGN
DEF *+4
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	argument	Double Real
z	result	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .CFER, .TSUB, .4ZRO, .ENTR

Notes:

1. Overflow will be set or cleared depending on occurrence. (Overflow only occurs if y >= 0 and x is the maximum negative number.)

2. .SIGN(x,0)=|x|

.SNCS

Purpose:	Calculate the sine or cosine of a real x (radians).		
Entry Points:	SIN		
Assembly:	DLD x JSB SIN Error return Return (result in A & B)		DLD x JSB COS Error return Return (result in A & B)
FORTRAN 4X:	Function:	SIN(x) COS(x)	
FORTRAN 77:	Function:	SIN(x) COS(x)	
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Real
Result:	Real in A & B		
Errors:	x outside [-8192*pi,+8191.75*pi] --> 05OR		
Program Type:	Type 6, Reentrant		
External References:	.ZPRV, .CMRS, ..FCM, .FMP, .FAD		

.SQRT

Purpose: Calculate the square root of a double real x.

Entry Points: .SQRT

Assembly:
```
JSB .SQRT
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: DSQRT (x) (with y option)
FORTRAN 77: Function: DSQRT (x) (with y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors: x < 0 gives error code 03 UN

Program Type: Type 7, Utility

External References: .ENTR, .CFER, .PWRZ, .TDJV, .XADD, .XDIV, .TADD, .SQRT

.TAN

Purpose: Calculate the tangent of a double real x (radians).

Entry Points: .TAN

Assembly:
```
JSB .TAN
DEF *+3
DEF y (result)
DEF x
Error return
Normal return
```

FORTRAN 4X: Function: DTAN (x) (with y option)
FORTRAN 77: Function: DTAN (x) (with y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors: x outside $[-2^{23}, 2^{23})$ --> 09 OR

Program Type: Type 7, Utility

External References: .ENTR, /CMRT, TRNL, .TDIV

.TANH

Purpose:	Calculate the hyperbolic tangent of a double real x.
Entry Points:	.TANH
Assembly:	JSB .TANH DEF *+3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: DTANH (with y option)
FORTRAN 77:	Function: DTANH (with y option)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result:	Double Real
Errors:	None
Program Type:	Type 7, Utility
External References:	.ENTR, .CFER, .TADD, .TDIV, /CMRT, /EXTH, .4ZRO

.TCPX

Purpose: Convert a double real x to a complex real y. The second value is set to zero.

Entry Points: .TCPX

Assembly:
```
JSB .TCPX
DEF y (result)
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Complex Real

Result: Complex Real

Errors: None

Program Type: Type 7, Utility

External References: .NGL

Notes: The result is rounded unless this would cause overflow. If so, overflow is set and the result is truncated to the greatest positive number.

.TENT

Purpose: Find the greatest integer i less than or equal to a double real (floor x).

Entry Points: .TENT

Assembly:
```
JSB .TENT
DEF *+3
DEF i (result)
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
i	result	Integer

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .FLUN, .ENTR, .CFER

Notes: Result is a double real value with no bits set after the binary point.

.TINT

Purpose:	Convert a double real x to an integer.
Entry Points:	.TINT, .TFXS
Assembly:	JSB .TINT DEF x Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result:	Integer in A
Errors:	None
Program Type:	Type 7, Utility
External References:	IFIX
Notes:	If the argument is outside the range $[-2^{15}, 2^{15})$, the result is $2^{15}-1$, and overflow is set. Overflow is cleared otherwise.

.TMTH

Purpose: Double real arithmetic.

Entry Points: .TADD, .TSUB, .TMPY, .TDIV

Assembly:

```
JSB .TADD    or   .TSUB    or   .TMPY    or .TDIV
DEF z (result)
DEF x
DEF y

z=x+y             z =x-y          z=x*y         z=x/y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	argument	Double Real
z	result	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .FLUN, .XFER, .CFER, FLOAT

Notes: If underflow occurs, zero is returned with overflow set. If overflow or divide by zero occurs, the largest positive number is returned with overflow set. Otherwise, overflow is cleared.

.TPWR

Purpose: Calculate x^i, where x is a double real and i is an unsigned integer.

Entry Points: .TPWR

Assembly:

```
LDA < i >
JSB .TPWR
DEF y (result)
DEF x
Return
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
i	exponent	Unsigned Integer
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .TMPY, FLOAT, .FLUN, .CFER

Notes:

1. "i" must be in the range [2,32768].
2. If overflow occurs, the maximum positive number is returned with overflow set. Overflow is set if underflow occurs.
3. The X- and Y-Registers may be altered.

.TSCS

Purpose: Calculate the sine or cosine of double precision x (radians).

Entry Points: .SIN, .COS

Assembly:

JSB SIN	or	JSB .COS
DEF *+3		DEF *+3
DEF y (result)		DEF y (result)
DEF x		DEF x
Error return		Error return
Normal return		Normal return

FORTRAN 4X: Function: DSIN (with y option)
DCOS (with y option)

FORTRAN 77: Function: DSIN (with y option)
DCOS (with y option)

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors: x outside $(-2^{23}, 2^{23})$ --> 05 OR

050R

Program Type: Type 7, Utility

External References: .SIN - .ENTR, /CMRT

.COS - TRNL, .TDIV

.TTOI

Purpose: Calculate x^i, where x is a double real and i is an integer.

Entry Points: .TTOI

Assembly:

```
JSB .TTOI
DEF y (result)
DEF x
DEF i
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
i	exponent	Integer
x	argument	Double Real
y	result	Double Real

Result: Double Real

Errors:

Condition	Error Code
x=0, I<=0	12 UN
$x^{\lvert I \rvert} >= 2^{127}$	(Floating point overflow)

Program Type: Type 7, Utility

External References: .TPWR, .TDIV, .CFER, .4ZRO

.TTOR

Purpose: Raise a double real x to a real power y.

Entry Points: .TTOR

Assembly:

```
JSB .TTOR
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	exponent	Real
z	result	Double Real

Result: Double Real

Errors: See .TTOT

Program Type: Type 6, Reentrant

External References: .TTOT

.TTOT

Purpose: Calculate x^y, where x and y are both double reals.

Entry Points: .TTOT

Assembly:

```
JSB .TTOT
DEF z (result)
DEF x
DEF y
Error return
Normal return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	exponent	Double Real
z	result	Double Real

Result: Double Real

Errors: See Notes.

Program Type: Type 7, Utility

External References: .LOG, .EXP, .CFER, .TMPY, .4ZRO

Notes:

1. Underflow will give a zero result, with no error. Overflow returns no result and gives an error code of 07 OF.
2. If (x<0) or (x=0 and y<=0), there will be an error code of 13 UN.

.XCOM

Purpose:	Complement a double real unpacked mantissa in place. Upon return, the A-Register = 1 if exponent should be adjusted; otherwise A = 0.
Entry Points:	.XCOM
Assembly:	JSB .XCOM DEF x ADA (exponent) STA (exponent) Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result:	Double Real
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV
Notes:	This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

.XDIV

Purpose:	Divide an extended real x by an extended real y.
Entry Points:	.XDIV
Assembly:	JSB .XDIV DEF z (result) DEF x DEF y Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	dividend	Extended Real
y	divisor	Extended Real
z	result	Extended Real

Result:	Extended Real
Errors:	See Notes.
Program Type:	Type 6, Reentrant
External References:	.ZRNT, .XPAK
Notes:	This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

.XFER

Purpose:	Move three words from address x to address y. Used for extended real transfers.
Entry Points:	.XFER
Assembly:	LDA (address of x) LDB (address of y) JSB .XFER Return (A = direct address of x+3) (B = direct address of y+3)
FORTRAN:	Not callable
Pascal:	Not callable
Result:	Extended Real
Errors:	None
Program Type:	Type 6, Privileged
External References:	.DFER, .ZPRV
Notes:	This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

.XMPY

Purpose: Multiply an extended real x by an extended real y.

Entry Points: .XMPY

Assembly:
```
JSB .XMPY
DEF z (result)
DEF x
DEF y
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real
y	argument	Extended Real
z	result	Extended Real

Result: Extended Real

Errors: See notes.

Program Type: Type 6, Privileged

External References: .XPAK, .ZPRV

Notes: This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description Chapter 1.

.XPAK

Purpose: Double real mantissa is normalized, rounded, and packed with exponent; result is a double real.

Entry Points: .XPAK

Assembly:

```
LDA exponent
JSB .XPAK
DEF x (3-word mantissa)
Return (result in x)
```

FORTRAN: Not callable

Pascal: Not callable

Result: Double Real

Errors: See XADSB

Program Type: Type 6, Privileged

External References: .ZPRV

Notes: This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.

If z is outside the range:

$$[-2^{128}, 2^{127}(1-2^{-39})],$$

then the overflow bit is set and

$$z = 2^{127}(1-2^{-39}).$$

If the result is within the range:

$$[-2^{-129}(1+2^{-22}), 2^{-129}],$$

then the overflow bit is set and z = 0.

.YINT

Purpose:	Truncate the fractional part of a double real.
Entry Points:	.YINT
Assembly:	JSB .YINT DEF *+3 DEF y (result) DEF x Return
FORTRAN 4X:	Function: DDINT (with y option)
FORTRAN 77:	Function: DDINT (with y option)
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Double Real

Result:	Double Real
Errors:	None
Program Type:	Type 7, Utility
External References:	.TENT, .TADD, .ENTR
Notes:	Result is a double real value with no bits set after the binary point.

..CCM

Purpose:	Complement a complex variable x in place.
Entry Points:	..CCM
Assembly:	JSB ..CCM DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Complex

Result:	Complex
Errors:	None
Program Type:	Type 7, Utility
External References:	..DLC

..DCM

Purpose: Extend a real complement in place.

Entry Points: ..DCM

Assembly:
```
JSB ..DCM
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real

Result: Extended Real

Errors: See Notes.

Program Type: Type 6, Reentrant

External References: .ZRNT, .XSUB

Notes:

1. This routine is available in 21MX Fast FORTRAN Processor (FFP) firmware. See the description in Chapter 1.
2. If x is the smallest negative number (-2^{127}), then result is the largest positive number [$(1-2^{-39}).2^{127}$] and the overflow bit is set.

..DLC

Purpose:	Load and complement a real x.
Entry Points:	..DLC
Assembly:	JSB ..DLC DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZPRV, .FSB

..FCM

Purpose:	Complement a real x.
Entry Points:	..FCM
Assembly:	DLD x JSB ..FCM Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result:	Real in A & B
Errors:	None
Program Type:	Type 6, Privileged
External References:	.ZRPV, .FSB

..TCM

Purpose: Negate a double real.

Entry Points: ..TCM

Assembly:

```
JSB ..TCM
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .TSUB, .4ZRO

Notes: This routine modifies the memory locations designated by x. Overflow is cleared unless x is:

$$-2^{127}$$

in which case overflow is set and x becomes $2^{127} -2^{82}$.

Chapter 3
Double Integer Subroutines

FIXDR

Purpose: Convert real to double length record number.

Entry Points: FIXDR

Assembly:

```
REAL x,y,FIXDR
   :
y=FIXDR(x)
Return
```

FORTRAN 4X: Function
FORTRAN 77: Function

Pascal: Not callable

Result: Double length record number (may be in real variable)

Errors: See notes.

Program Type: Type 7, Utility

External References: .FIXD, .ENTR

Notes: Result is incorrect if real value is greater than $2^{31}-1$ since this is the maximum record number. Record numbers greater than 2^{23} cannot be represented exactly as real numbers.

FLTDR

Purpose: Convert a double length record number to real.

Entry Points: FLTDR

Assembly:
```
DLD
JSB FLTDR
DEF *1
DST real
```

FORTRAN 4X: Function
FORTRAN 77: Function

Pascal: Not callable

Result: Real

Errors: None

Program Type: Type 7, Utility

External References: .FLTD, .ENTR

Notes: Should not be used for record numbers exceeding 2^{23}, as the conversion may not be exact for such numbers.

.DADS

Purpose: Double integer add and subtract.

Entry Points: .DAD, .DSB, .DSBR

Assembly:

```
DLD x                DLD x             DLD x
JSB .DAD             JSB .DSB          JSB .DSBR
DEF y                DEF y             DEF y
<----------Return (results in A & B) ------------>
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer
y	argument	Double Integer

Result: Double Integer

Errors: None

Program Type: Type 7, Utility

External References: None

Notes: If overflow occurs, the least significant 32 bits are returned with overflow set. Overflow is cleared otherwise. "E" is never cleared, but is set if carry occurs (.DAD) or borrow (.DSB & DSBR).

.DSBR is used to replace the sequence:

```
DSD temp                          JSB .DSBR
DLD x           with              DEF x
JSB .DSB
DEF temp
```

.DCO

Purpose: Compare two double integers.

Entry Points: .DCO

Assembly:

```
DLD x
JSB .DCO
DEF y
Return (if x=y)
Return (if x<y)
Return (if x>y)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer
y	argument	Double Integer

Result: None

Errors: None

Program Type: Type 7, Utility

External References: None

Notes: A, B, E & O, are left unchanged. The compare is correct even if X-Y is not representable in 32 bits.

.DDE

Purpose:	Decrement the double integer in the A- & B-Registers.
Entry Points:	.DDE
Assembly:	DLD x JSB .DDE Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer

Result:	Double Integer
Errors:	None
Program Type:	Type 7, Utility
External References:	None

Notes:

1. If the largest negative number is decremented, the largest positive number is the result, with overflow set. Overflow is cleared otherwise.

2. "E" is preserved unless X=0, in which case it is set.

.DDI

Purpose: Double integer divide.

Entry Points: .DDI, ,DDIR

Assembly:

```
DLD x                          DLD y
JSB .DDI                       JSB .DDIR
DEF y                          DEF x
Return (result in A & B)       Return (result in A & B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	dividend	Double Integer
y	divisor	Double Integer

Result: Double Integer

Errors: None

Program Type: Type 7, Utility

External References: FLOAT

Notes: If overflow or divide by zero occurs, the largest positive integer is returned with overflow set. Overflow is cleared otherwise. "E" is preserved.

.DDIR is used to replace the sequence:

```
DSD temp      with      JSB .DDIR
DLD x                   DEF x
JSB .DDI
DEF temp
```

.DDS

Purpose: Double integer decrement and skip if zero.

Entry Points: .DDS

Assembly:

```
JSB .DDS
DEF x
Return (if x-1 not equal to 0)
Return (if x-1 equal to 0)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer

Result: Double Integer

Errors: None

Program Type: Type 7, Utility

External References: None

Notes: This routine decrements the double integer x. A, B, E & O, are left unchanged except that A & B will be changed if the effective address is zero.

.DIN

Purpose: Increment the double integer in the A- & B-Registers.

Entry Points: .DIN

Assembly:
```
DLD x
JSB .DIN
Return (result in A & B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer

Result: Double Integer

Errors: None

Program Type: Type 7, Utility

External References: None

Notes: If the largest positive number is incremented, the largest negative number is the result, with overflow set. Overflow is cleared otherwise. "E" is preserved unless X = -1, in which case "E" is set.

.DIS

Purpose:	Double integer increment and skip if zero.		
Entry Points:	.DIS		
Assembly:	JSB .DIS DEF x Return (if x+1 not equal to 0) Return (if x+1 equal to 0)		
FORTRAN:	Not callable		
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Double Integer
Result:	Double Integer		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	None		
Notes:	This routine increments the double integer x by 1. A, B, E & O are left unchanged except that A & B will be changed if the effective address is zero.		

.DMP

Purpose:	Double integer multiply.		
Entry Points:	.DMP		
Assembly:	DLD X JSB .DMP DEF y Return (result in A & B)		
FORTRAN:	Not callable		
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Double Real
	y	argument	Double Real
Result:	Double Integer		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	None		
Notes:	If overflow occurs, the largest positive integer is returned with overflow set. Overflow is cleared otherwise. "E" is preserved.		

.DNG

Purpose:	Negate double integer x.		
Entry Points:	.DNG		
Assembly:	DLD x JSB .DNG Return (result in A & B)		
FORTRAN:	Not callable		
Pascal:	Not callable		
Parameters:	Parameter	Description	Type
	x	argument	Double Real
Result:	Double Integer		
Errors:	None		
Program Type:	Type 7, Utility		
External References:	None		
Notes:	If overflow occurs, the argument is returned unchanged and overflow is set. Overflow is cleared otherwise. "E" is preserved unless X=0, in which case E=1.		

.FIXD

Purpose: Convert real to double integer.

Entry Points: .FIXD

Assembly:

```
DLD x
JSB .FIXD
Return (result in A & B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Real

Result: Double Integer in A & B

Errors: None

Program Type: Type 7, Utility

External References: .FLUN

Notes:

1. If the argument is outside the range $[-2^{31}, 2^{31})$ the result is $2^{31}-1$ and overflow is set. Overflow is cleared otherwise.

2. .FXDE is not a usable entry point. It is referenced by .XFXD and .TFXD.

.FLTD

Purpose:	Convert double integer to real.	
Entry Points:	.FLTD	
Assembly:	DLD x JSB .FLTD Return (result in A & B)	
FORTRAN:	Not callable	
Pascal:	Not callable	

Parameters:

Parameter	Description	Type
x	argument	Double Integer

Result:	Real in A & B
Errors:	None
Program Type:	Type 7, Reentrant
External References:	.PACK
Notes:	If the argument is outside the range $[-2^{23}, 2^{23})$ the excess low-order bits are truncated. Positive numbers may become smaller, negative numbers may become smaller in value (larger in absolute value).

.TFTD

Purpose: Convert a double integer to a double real.

Entry Points: .TFTD

Assembly:

```
DLD x
JSB .TFTD
DEF y (result)
Return
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Integer
y	result	Double Real

Result: Double Real

Errors: None

Program Type: Type 7, Utility

External References: .XPAK

.TFXD

Purpose:	Convert a double real to a double integer.
Entry Points:	.TFXD
Assembly:	JSB .TFXD DEF x Return (result in A & B)
FORTRAN:	Not callable
Pascal:	Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Real

Result:	Double Integer in A & B
Errors:	None
Program Type:	Type 7, Utility
External References:	.FLUN, .CFER, .FIXD, .FXDE
Notes:	If the argument is outside the range $[-2^{31}, 2^{31})$ the result is $2^{31}-1$ and overflow is set. Overflow is cleared otherwise.

.XFTD

Purpose: Convert a double integer to an extended real.

Entry Points: .XFTD

Assembly:

```
DLD x
JSB .XFTD
DEF y (result)
Return
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Double Real
y	result	Extended Real

Result: Extended Real

Errors: None

Program Type: Type 7, Utility

External References: .XPAK

.XFXD

Purpose: Convert extended real to double integer.

Entry Points: .XFXD

Assembly:
```
JSB .XFXD
DEF x
Return (result in A & B)
```

FORTRAN: Not callable

Pascal: Not callable

Parameters:

Parameter	Description	Type
x	argument	Extended Real

Result: Double Integer in A & B

Errors: None

Program Type: Type 7, Utility

External References: .FLUN, .FIXD, .FXDE, .XFER

Notes: If the argument is outside the range $[-2^{31}, 2^{31})$ the result is $2^{31}-1$ and overflow is set. Otherwise, overflow is cleared.

Chapter 4
Utility Subroutines

ABREG

Purpose:	FORTRAN A- and B-Register GET Routine.		
Entry Points:	ABREG		
Assembly:	JSB ABREG DEF *+3 DEF IA DEF IB Return		
FORTRAN 4X:	Call ABREG (IA,IB)		
FORTRAN 77:	Call ABREG (IA,IB)		
Pascal:	Callable:	IA <-- AREG IB <-- BREG	
Errors:	None		
Program TYPE:	Type 7, Utility		
External References:	None		

CLRIO

Purpose:	Dummy compatibility routine for use by the FORTRAN compilers (was used by BCS system).
Entry Points:	CLRIO
Assembly:	JSB CLRIO DEF *+1 Return (All registers remain intact.)
FORTRAN 4X:	Call CLRIO
FORTRAN 77:	Call CLRIO
Pascal:	Callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

ER0.E

Purpose: Specify the LU for printing library error messages. ER0.E is defaulted to 0.

Entry Points: None

Assembly:

```
EXT ER0.E
 .
 .
 .
LDA LU
STA ER0.E
Return
```

FORTRAN: Not callable

Pascal: Not callable

Result: None

Errors: None

Program TYPE: Type 7, Utility

External References: None

ERR0

Purpose:	Print a 4-character error code and a memory address on the logical unit ER0.E.
Entry Points:	ERR0
Assembly:	LDA nn LDB xx JSB ERR0 Return
FORTRAN:	Not callable
Pascal:	Not callable
Result:	Printed
Errors:	None
Program TYPE:	Type 7, Utility
External References:	REIO, ER0.E, PNAME

GETAD

Purpose:	Determine the true address of a parameter passed to a subroutine and place the address in ADRES.
Entry Points:	GETAD, ADRES
Assembly:	JSB GETAD DEF SUB,i LDA ADRES Return
FORTRAN:	Not callable
Pascal:	Not callable
Result:	GETAD <-- Address ADRES <-- Integer
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None
Notes:	May not be called by privileged or re-entrant routines; refer to .PCAU.

IGET

Purpose:	Allow programs to read the contents of a memory address.
Entry Points:	IGET
Assembly:	JSB IGET DEF *+2 DEF IADRS Return (results in A)
FORTRAN:	Callable as a function
Pascal:	Callable
Result:	Contents of memory address.
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None
Notes:	This routine is for FORTRAN users only.

IND.E

Purpose:	Used by .INDR and .INDA routines to select output LU for error messages. Default is 6; a 0 inhibits messages.
Entry Points:	IND.E
Assembly:	EXT IND.E LDA LU STA IND.E Return
FORTRAN:	Not callable
Pascal:	Not callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

ISSR

Purpose: Set the S-Register to the value n.

Entry Points: ISSR

Assembly:
```
JSB ISSR
DEF *+2
DEF n
Return
```

FORTRAN 4X: Call ISSR(n)
FORTRAN 77: Call ISSR(n)

Pascal: Callable

Result: None

Errors: None

Program TYPE: Type 7, Utility

External References: None

ISSW

Purpose:	Set the sign bit (15) of A-Register equal to bit n of the switch register.
Entry Points:	ISSW
Assembly:	LDA n JSB ISSW Return (result in A)
FORTRAN 4X:	Function: ISSW(n)
FORTRAN 77:	Function: ISSW(n)
Pascal:	Not callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

MAGTP

Purpose: Performs utility functions on magnetic tape and other devices: checks status, performs rewind/standby, writes a gap, and issues a clear request.

Entry Points: IOEF, IERR, IEOT, ISOT, LOCAL,IWRDS(N/A in RTE),RWSTB

Assembly: The calling sequence and purpose of each entry point is:

JSB IEOF DEF *+2 DEF unit Return	Returns a negative value in A if an end-of-file was encountered during last tape operation on the logical unit specified.
JSB IERR DEF *+2 DEF unit Return	Returns a negative value in A if a parity or timing error was not cleared after three read attempts during the last operation on the specified unit (cannot occur if EOF occurs).
JSB IEOT DEF *+2 DEF unit Return	Returns a negative value in A if an end-of-tape was encountered during the last forward movement of the specified unit.
JSB ISOT DEF *+2 DEF unit Return	Returns a negative value in A if the start-of-tape marker is under the tape head of the specified unit.
JSB LOCAL DEF *+2 DEF unit Return	Returns a negative value in A if the specified unit is in local mode.
JSB IWRDS DEF *+2 DEF unit Return	(Not available in RTE.) Returns the value of the transmission log of the last read/write operation on the specified unit. (In the formatter environment, this value is always a positive number of characters.)
JSB RWSTB DEF *+2 DEF unit Return	Rewinds the specified logical unit and sets it to LOCAL.

FORTRAN 4X:	Callable as a subroutine
FORTRAN 77:	Callable as a subroutine
Pascal:	Callable
Result:	Not Applicable
Errors:	Returns on illegal call
Program TYPE:	Type 7, Utility
External References:	.ENTR, EXEC

NAMR

Purpose: Read an input buffer of any length and produce a parameter buffer of 10 words.

Entry Points: NAMR

Assembly:

```
JSB NAMR
DEF *+5
DEF IPBUF
DEF INBUF
DEF LENGTH
DEF ISTRC
Return
```

NAMR = -1 if no characters are in INBUF.
NAMR = 0 if the character string has been parsed. (see note)

where: IPBUF = 10 word destination parameter buffer.

The ten words are described as follows:

Word 1 = 0 if type = 0 (see below)
= 16-bit number if type = 1. If number is negative, number is in two's complement.
= Chars 1 & 2 if type = 3.

Word 2 = 0 if type =0 or 1, chars 2 & 3 or trailing space(s) if 3.

Word 3 = Same as word 2. (Type 3 parameters is left justified.)

Word 4 = Parameter type of all 7 parameters in 2 bit pairs. Note the difference between NAMR parameter types, and those for the system library routine PARSE.

0 = Null parameter.
1 = Integer numeric parameter.
2 = Not implemented yet. (FMGR?)
3 = Left-justified 6 ASCII character parameter.

Bits for	,FNAME	: P1	: P2	: P3	: P4	: P5	: P6,
	0,1	2,3	4,5	6,7	8,9	10,11	12,13

Word 5 = 1st sub-parameter and has characteristics of word 1.

Word 6 = 2nd sub-parameter delimited by colons as in word 5.

Word 7 = 3rd sub-parameter as 5 & 6. (May be 0, number or 2 chars.)

Word 8 = 4th " " "

Word 9 = 5th " " "

Word 10 = 6th sub-parameter. (For possible futures i.e., system #)

INBUF = Starting address of input buffer containing "NAMR".

LENGTH = Character length of INBUF.

ISTRC = Starting character number in INBUF. This parameter will be updated for possible next call to NAMR and the start character in INBUF.

Caution: ISTRC is modified by this routine, therefore, it must be passed as a variable (not a constant) from caller (FTN).

FORTRAN 4X: Callable: If (NAMR (IPBUF,INBUF,LENTH,ISTRC)) 10,20
FORTRAN 77: Callable: If (NAMR (IPBUF,INBUF,LENTH,ISTRC)) 10,20

Pascal: Callable

Result: None

Errors: None

Program TYPE: Type 7, Utility

External References: .ENTR

Notes: Examples that can be parsed:

+12345, DOUG:DB:-12B:,,GEORGE: A,

&PARSE:JB::4:-1:1775:123456B

where:

NAMR#	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10
1	12345	0	0	00001B	0	0	0	0	0	
2	DO	UG		00037B	DB	-10	0	0	0	0
3	0	0	0	00000B	0	0	0	0	0	0
4	GE	OR	GE	00017B	A	0	0	0	0	0
5	$P	AR	SE	12517B	JB	0	4	-1	1775	-22738

TEST PROGRAM

```
FTN,L
      PROGRAM TESTN
      DIMENSION IB(36),IDMY(2),IPBUF(10)
      EQUIVALENCE (IDMY,DMY),(LEN,IDMY(2))
    1 WRITE (1,100)
  100 FORMAT ("INPUT ASCII NAMR'S TO PARSE?")
      DMY = EXEC (1,401B,IB,-72)
      ISCR = 1
      DO 200 I=1,10
      IF (NAMR(IPBUF,IB,LEN,ISCR)) 1,210
  210 WRITE (1,220) ISCR,IPBUF,IPBUF
  220 FORMAT (" "/,I3,10(X,I6)/" "3A2,7(X,06))
  200 CONTINUE
      STOP
      END
      END$
```

OVF

Purpose:	Return value of overflow bit in bit 15 of the A-Register and clear the overflow bit.
Entry Points:	OVF
Assembly:	JSB OVF DEF RTN Return (result in A)
FORTRAN 4X:	Callable (see notes)
FORTRAN 77:	Callable (see notes)
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None
Notes:	IF (OVF(IDMY)) 10,20 10 - start of user's overflow set routine. 20 - start of user's overflow clear routine.

PAU.E

Purpose:	Used by .PAUS and .STOP routines to select LU on which to output pause message. Default is 1; a 0 inhibits message.
Entry Points:	PAU.E
Assembly:	EXT PAU.E LDA LU STA PAU.E Return
FORTRAN:	Not callable
Pascal:	Not callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

PAUSE

Purpose:	Print the following message on the console device: name: PAUSE xxxxx where name is the calling program name and xxxxx is the specified integer i. Halt program execution and return to operating system.
Entry Points:	.PAUS, .STOP
Assembly:	LDA i JSB .PAUS (or .STOP) Return (see notes)
FORTRAN:	Not callable
Pascal:	Not callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	EXEC, PAU.E, REIO, PNAME
Notes:	When .PAUS is used, the program may be continued using GO (RTE) or :GO (DOS).

PNAME

Purpose: Moves the name of the currently executing program from the program's ID segment to a three word array.

Entry Points: PNAME

Assembly:
```
JSB PNAME
DEF *+2
DEF IARAY
-->
IARAY BSS 3
Return
```

FORTRAN 4X: (CALL PNAME (IARAY))
FORTRAN 77: (CALL PNAME (IARAY))

Pascal: Callable

Result: ASCII characters

Errors: None

Program TYPE: Type 7, Utility

External References: .ENTR, $OPSY

Notes: The sixth character is returned as an ASCII space.

<u>Sample Program</u>
```
      PROGRAM PRNAM
      DIMENSION IARAY(3)
      CALL PNAME (IARAY)
      WRITE (1,100) IARAY
  100 FORMAT ("  PROGRAM  ",3A2,"EXECUTING:")
      STOP     ^^         ^
```

PTAPE

Purpose: Position a magnetic tape unit by spacing forward or backward a number of files and/or records.

Entry Points: PTAPE

Assembly:

```
JSB PTAPE
DEF *+4
DEF logical unit
DEF file count     (see notes)
DEF record count       "
Return
```

For example:

 0 = make no file movements.

-1 = backspace to the beginning of the current file.

 1 = forward space to the beginning of the next file.

-2 = backspace to the beginning of the previous file.

Record count: positive for forward, negative for backward.

The file count is executed first, then the record count. EOF marks count as a record.

For example:

0,-1 = move back one record.

-1,0 = backspace to the first record of the current file.

See notes.

FORTRAN 4X: Call PTAPE (LU,file count,record count)
FORTRAN 77: Call PTAPE (LU,file count,record count)

Pascal:	Callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	EXEC, .ENTR
Notes:	After using PTAPE, always check status with MAGTP.

RMPAR

Purpose: Move five parameters from the programs ID segment into a buffer within the program memory space. If the program resides in a partition, the parameters are cross loaded from the system maps. Used to retrieve up to five parameters passed to a program by the operating system (see notes).

Entry Points: RMPAR

Assembly:
```
Suspend call or program
        entry point
     JSB RMPAR
     DEF *+2
     DEF ARRAY
     -->
ARRAY BSS 5
Return
```

FORTRAN 4X: Callable
FORTRAN 77: Callable

Pascal: Callable

Result: Integer

Errors: None

Program TYPE: Type 7, Utility

External References: $OPSY

Notes:

1. The operating system will insert parameters into a program's ID segment as a result of:

   a. ON, GO, and other functions in RTE (refer to RTE manual for other functions of this call).

   b. Program execution of an EXEC call.

2. The RMPAR call must occur as the first executable instruction in the program or as the first executable instruction following the program suspend call.

   Example:

```
FTN,L
      PROGRAM TEST
      DIMENSION IBUF (5)
      CALL RMPAR (IBUF)
             or
      PAUSE
      CALL RMPAR (IBUF)
```

SREAD

Purpose: Read a source record or sector from a device specified by a logical unit number. (Used only by system programs.)

Entry Points: %READ, %JFIL, %RDSC, (see notes)

Assembly:

```
JSB %READ
DEF *+5
DEF input logical unit
DEF input buffer
DEF negative number of characters
EOP return
Return (B = number of characters)

LDA code
LDB sector #
JSB %RDSC
Return (A = last word in sector)

JSB %JFIL
Return (A = last word in sector)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: None

Errors: None

Program TYPE: Type 7, Utility

External References: $OPSY, EXEC

Notes: Entry Points:

%READ - reads a source record from disc or other device specified by logical unit number.
%RDSC - reads a specified sector, returning the (RTE) code word.
%JFIL - rewinds source; reads sector pointed to by the base page source-file code word.

#COS

Purpose:	Calculate the cosine with no error return.
Entry Points:	#COS
Assembly:	JSB #COS DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Complex
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERR0, .ENTR, CCOS

#EXP

Purpose:	Entry to CEXP with no error return.
Entry Points:	#EXP
Assembly:	JSB #EXP DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Complex
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERR0, .ENTR, CEXP

#LOG

Purpose:	Entry to CLOG with no error return.
Entry Points:	#LOG
Assembly:	JSB #LOG DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Complex
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERR0, .ENTR, CLOG

#SIN

Purpose:	Calculate the sine with no error return.
Entry Points:	#SIN
Assembly:	JSB #SIN DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Complex
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERRO, .ENTR, CSIN

$EXP

Purpose:	Entry to DEXP with no error return.
Entry Points:	$EXP
Assembly:	JSB $EXP DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Extended Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERR0, .ENTR, DEXP

$LOG

Purpose:	Entry to DLOG with no error return.
Entry Points:	$LOG
Assembly:	JSB $LOG DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Extended Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ERR0, .ENTR, DLOG

$LOGT

Purpose:	Entry to DLOGT with no error return.
Entry Points:	$LOGT, $LOGO
Assembly:	JSB $LOGT (or LOGO) DEF *+3 DEF y DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	DLOGT, .ENTR, ERRO

$SETP

Purpose: Set up a list of pointers.

Entry Points: $SETP

Assembly:

```
LDA <starting pointer>
LDB <starting address to be set>
JSB $SETP
DEF <count>
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Integer

Errors: None

Program TYPE: Type 6, Re-entrant

External References: .ZPRV

Notes:

1. This routine is available in microcode.

2. The sign bit of B is ignored.

$SQRT

Purpose: Entry to DSQRT with no error return.

Entry Points: $SQRT

Assembly:
```
JSB $SQRT
DEF *+3
DEF y
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Extended Real

Errors: None

Program TYPE: Type 7, Utility

External References: DSQRT, ERR0, .ENTR

%ABS

Purpose: Call-by-name entry to IABS(i).

Entry Points: %ABS

Assembly:

```
JSB %ABS
DEF *+2
DEF i
Return (result in A)
```


FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Integer in A

Errors: None

Program TYPE: Type 7, Utility

External References: IABS

$TAN

Purpose: DTAN with no error return.

Entry
Points: $TAN

Assembly:
```
JSB DTAN
DEF *+3
DEF y
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Double Real

Errors: x outside [-8192*pi,+8192.75*pi] --> 09 OR

Program
TYPE: Type 7, Utility

External
References: DTAN, .ENTR

%AN

Purpose:	Call-by-name entry to TAN(x).
Entry Points:	%AN
Assembly:	JSB %AN DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	None
Program TYPE:	Type 7, Utility
External References:	TAN, ERR0

%AND

Purpose:	Call-by-name entry to calculate the logical "and" (product) of the two integers i and j.
Entry Points:	%AND
Assembly:	JSB %AND DEF *+3 DEF i DEF j Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

%ANH

Purpose:	Call-by-name entry to TANH(x)
Entry Points:	%ANH
Assembly:	JSB %ANH DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	None
Program TYPE:	Type 7, Utility
External References:	TANH

%BS

Purpose: Call-by-name entry to ABS(x).

Entry
Points: %BS

Assembly:
```
JSB %BS
DEF *+2
DEF x
Return (result in A & B)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Real in A & B

Errors: None

Program
TYPE: Type 7, Utility

External
References: ABS

%FIX

Purpose:	Call-by-name entry to IFIX(x).
Entry Points:	%FIX
Assembly:	JSB %FIX DEF *+2 DEF x Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	IFIX

%IGN

Purpose: Call-by-name entry to SIGN (x,z)

Entry Points: %IGN

Assembly:
```
JSB %IGN
DEF *+3
DEF x
DEF z
Return (result in A & B)
```

FORTRAN 4X: Not callable

FORTRAN 77: Callable

Pascal: Callable

Result: Real

Errors: None

Program TYPE: Type 7, Utility

External References: SIGN

%IN

Purpose:	Call-by-name entry to SIN(x).
Entry Points:	%IN
Assembly:	JSB %IN DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	See SIN
Program TYPE:	Type 7, Utility
External References:	SIN, ERR0

%INT

Purpose:	Call-by-name entry to AINT(x).
Entry Points:	%INT
Assembly:	JSB %INT DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	AINT

%LOAT

Purpose:	Call-by-name entry to FLOAT (I).
Entry Points:	%LOAT
Assembly:	JSB %LOAT DEF *+2 DEF I Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	None
Program TYPE:	Type 7, Utility
External References:	FLOAT

%LOG

Purpose: Call-by-name entry to ALOG(x).

Entry Points: %LOG

Assembly:
```
JSB %LOG
DEF *+2
DEF x
Return (result in A & B)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Real in A & B

Errors: See ALOG

Program TYPE: Type 7, Utility

External References: ALOG, ERRO

%LOGT

Purpose:	Call-by-name entry to ALOGT (x).
Entry Points:	%LOGT, %LOGO
Assembly:	JSB %LOGT (or %LOGO) DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ALOGT, ERRO

%NT

Purpose:	Call-by-name entry to INT (x).
Entry Points:	%NT
Assembly:	JSB %NT DEF *+2 DEF x (real) Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer
Errors:	None
Program TYPE:	Type 7, Utility
External References:	INT

%OR

Purpose:	Call-by-name entry to calculate the inclusive "OR" of two integers, i and j.
Entry Points:	%OR
Assembly:	JSB %OR DEF *+3 DEF i DEF j Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

%OS

Purpose:	Call-by-name entry to COS (x).
Entry Points:	%OS
Assembly:	JSB %OS DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	See COS
Program TYPE:	Type 7, Utility
External References:	COS, ERR0

%OT

Purpose:	Standard call-by-name subroutine for NOT function.
Entry Points:	%OT
Assembly:	JSB %OT DEF *+2 DEF i Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

%QRT

Purpose:	Call-by-name entry to SQRT (x).
Entry Points:	%QRT
Assembly:	JSB %QRT DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	See SQRT
Program TYPE:	Type 7, Utility
External References:	SQRT, ERR0

%SIGN

Purpose:	Call-by-name entry to ISIGN (i,z).
Entry Points:	%SIGN
Assembly:	JSB %SIGN DEF *+3 DEF i DEF z Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ISIGN

%SSW

Purpose:	Call-by-name entry to ISSW (x).
Entry Points:	%SSW
Assembly:	JSB %SSW DEF *+2 DEF n (integer) Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	ISSW

%TAN

Purpose: Call-by-name entry to ATAN (x).

Entry Points: %TAN

Assembly:
```
JSB %TAN
DEF *+2
DEF x
Return (result in A & B)
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Real in A & B

Errors: See ATAN

Program TYPE: Type 7, Utility

External References: ATAN, ERRO

%WRIS

Purpose:	Writes a disc source file (used only by system programs).
Entry Points:	%WRIS, %WRIN, %WEOF
Assembly:	None
FORTRAN:	Not callable
Pascal:	Not callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	EXEC
Notes:	This routine can only be called in the RTE System.

%WRIT

Purpose:	Writes a load-and-go file on disc (used only by system programs).
Entry Points:	%WRIT, %WRIF, %WBUF
Assembly:	None
FORTRAN:	Not callable
Pascal:	Not callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	$OPSY, EXEC

%XP

Purpose:	Call-by-name entry to EXP (x).
Entry Points:	%XP
Assembly:	JSB %XP DEF *+2 DEF x Return (result in A & B)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Real in A & B
Errors:	See EXP
Program TYPE:	Type 7, Utility
External References:	EXP, ERR0

.ENTC

Purpose: Transfer the true address of parameters from a calling sequence into a subroutine and adjust return addresses to the true return point.

Entry Points: .ENTC

Assembly: Same as .ENTP
.ENTR
Return

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Address

Errors: None

Program TYPE: Type 6, Privileged

External References: .ZPRV

Notes: This routine assumes the subroutine call is of the form:

JSB SUB
DEF p_1 (first parameter)
.
.
.
DEF p_m

The number of parameter addresses actually passed by the calling routine must agree with the number requested by the receiving routine.

.ENTR

Purpose: Transfer the true addresses of the parameters from a calling sequence into a subroutine; adjust return address to the true return point.

Entry Points: .ENTR, .ENTP

Assembly: For all utility routines:

```
PARAM BSS n     (n = maximum # of parameters)
  SUB NOP       (entry point to subroutine)
      JSB .ENTR
      DEF PARAM             * See notes.
      Return
```

For all privileged routines:

```
PARAM BSS n     (n = maximum # of parameters)
  SUB NOP       Subroutine entry point
      JSB .ZPRV
      DEF LIBX
      JSB .ENTP
      DEF PARAM
       :
 LIBX JMP SUB,I
      DEF LIBX
      Return
```

For all re-entrant routines:

```
      TDB NOP        (re-entrant processing table)
          DEC Q+N+3  (size of table)
          NOP
          BSS Q      (subroutine variables)
    PARAM BSS n      (n = maximum # of parameters)
          SUB NOP    (subroutine entry point)
          JSB .ZRNT
          DEF LIBX
          JSB .ENTP
          DEF PARAM
          STA TBD+2  (return address)
           :
     LIBX JMP TDB+2,I
          DEF TDB
          DEC 0
          Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Address

Errors: None

Program TYPE: Type 6, Privileged

External References: .ZPRV

Notes:

1. The true parameter address is determined by eliminating all indirect references.

2. .ENTR and .ENTP assume the subroutine call is of the form:

```
JSB SUB
DEF *+m+1    (m = number of parameters)
DEF p
 .   1
 .
 .
DEF p
     m
```

   If m > n, then n parameters will be passed. If n > m, then m parameters will be passed, and any parameter addresses not passed remain as they were from the previous call.

3. "PARAM BSS n" must appear immediately before the subroutine entry point "SUB NOP". The entry point is set to the return address (DEF *+m+1). "JSB .ENTR" must be the first instruction after the subroutine entry point. "JSB .ENTP" must be the third instruction after the subroutine entry point.

4. This routine is available in 21MX FFP firmware. See Chapter 1.

.FMUI, .FMUO, .FMUP

Purpose: .FMUI contains three entry points corresponding to three conversion procedures:

.FMUI - Convert an ASCII digit string to internal numeric form.

.FMUO - Convert a numeric value to ASCII.

.FMUP - Convert an unpacked internal format number (from .FMUI) to a normal format.

Entry Points: .FMUI, .FMUO, .FMUP

Assembly:

```
JSB .FMUI
DEF *+8
DEF <buffer> ASCII, one digit/word, FORTRAN R1 format
DEF <bufsiz> # of digits in <buffer> between 0 and 20,
             inclusive.
DEF <sign>   0 = positive, 1 = negative.
DEF <exp>    scale factor; power of ten.
DEF <result> return value.
DEF <type>   type of <result> (see below).
DEF <ovfl>   returned from .FMUI, 1 if overflow or
             underflow else 0.
Return

JSB .FMUO
DEF *+7
DEF <buffer> returned from .FMUO
DEF <bufsiz> returned from .FMUO
DEF <sign>   returned from .FMUO
DEF <exp>    returned from .FMUO
DEF <value>  input value
DEF <type>   type of value (see below)
Return

JSB .FMUP
DEF *+5
DEF <result>
DEF <type>
DEF <unpkd>  input <result> from .FMUI
DEF <ovfl>   returned from .FMUP, 1 if overflow or
             underflow else 0.
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: None

Errors: None

Program Type: Type 7, Utility

External References: .PACK, .ENTR, .MVW, IFIX

Method:

.FMUI - The value in <buffer> is converted to binary with the digit in buffer (i) having weight 10** (<exp>-i). As if it were of the form 0. <buffer> E <exp>. The result is negated if <sign> =1, and rounded to the specified type:

<TYPE> =	TYPE
0	16-bit integer (1 word)
1	32-bit integer (2 words)
2	32-bit Real (2 words)
3	48-bit Real (3 words)
4	64-bit Real (4 words)
5	unpacked internal format (5 words)

.FMUO - Reverse of .FMUI, i.e., generates <buffer>, <exp>, and <sign> from <value> as described in .FMUI. The result should be rounded by calling .FMUR since there may be some round-off error by .FMUO such as 2.0 may convert to 1.99999.

.FMUP - A type 5 buffer <unpkd> created by .FMUI is converted to a normal type buffer <result>. The type of <result> is specified by <type> and must be 0 to 4.

.FMUR

Purpose: Rounding of digit string produced by .FMUO.

Entry Points: .FMUR

Assembly:

```
JSB .FMUR
DEF *+5
DEF <buffer> ASCII, one digit/word, FORTRAN R1 format
             (input and returned value)
DEF <bufsiz> # of digits in <buffer> between 0 and
             20, inclusive
DEF <rndsiz> # of digits to round to
DEF <ovfl>   returned from .FMUR, 1 if carry overflow
             occurs, else 0.
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Example:

A conversion to 10 digits would be as follows:

```
.FMUO (buffer,11,sign,exp,value,type)
.FMUR (buffer,11,10,ovfl)
exp=exp+ovfl
```

Result: None

Errors: None

Program Type: Type 7, Utility

External References: .ENTR

.GOTO

Purpose: Transfer control to the location indicated by a FORTRAN computed GOTO statement:

$$GOTO\ (k_1, k_2, \ldots k_n)\ j$$

Entry Points: .GOTO

Assembly:
```
JSB .GOTO
DEF *+n+2
DEF J
DEF k1
 .
 .
 .
DEF kn
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Branch to address k_j

Errors: If $j < 1$ then k_1; if $j>n$ then k_n

Program TYPE: Type 7, Utility

External References: None

Notes: This routine is available in 21MX FFP firmware. See Chapter 1.

.MAP

Purpose:	Return actual address of a particular element of a two-dimensional FORTRAN array.
Entry Points:	.MAP.
Assembly:	JSB .MAP. DEF array DEF first subscript DEF second subscript OCT first dimension Return (result in A)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Integer in A
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None

.OPSY

Purpose: Determines which operating system is in control. Included for compatibility with previous libraries.

Entry Points: .OPSY

Assembly:

```
JSB .OPSY
--> result in A
A = -7  (RTE-MI)
A = -15 (RTE-MII)
A = -5  (RTE-MIII)
A = -3  (RTE-II)
A = -1  (RTE-III)
A = -9  (RTE-IV)
A = -17 (RTE-6/VM)
A =  1  (DOS)
Return
```

FORTRAN 4X: Not callable

FORTRAN 77: Callable

Pascal: Callable

Result: Integer in A

Errors: None

Program TYPE: Type 7, Utility

External References: $OPSY

Notes: This routine is equivalent to:

```
EXT $OPSY
LDA $OPSY
```

.PCAD

Purpose:	Return the true address of a parameter passed to a subroutine.
Entry Points:	.PCAD
Assembly:	JSB .PCAD DEF SUB,i Return (result in A) (see notes)
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Direct address: A
Errors:	None
Program TYPE:	Type 6, Privileged
External References:	.ZPRV

Notes:

1. .PCAD has the same purpose as GETAD.

2. .PCAD is used by re-entrant or privileged subroutines because they cannot use GETAD.

.RCNG

Purpose:	Convert calls using .ENTR to .ENTC convention.
Entry Points:	.RCNG
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None
Notes:	The subroutine subr is one of the seven non-intrinsic entry points:
	XADD, XSUB, XDIV, CADD, CSUB, CDIV, CMPY

.TAPE

Purpose:	Perform magnetic tape rewind, backspace or end-of-file operations on a specified logical unit.
Entry Points:	.TAPE
Assembly:	LDA constant JSB .TAPE Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	None
Errors:	None
Program TYPE:	Type 7, Utility
External References:	EXEC
Notes:	In FORTRAN use utility statements or PTAPE and MGTAP.

..MAP

Purpose: Computes the address of a specified element of a 1, or 2 or 3 dimension array; returns the address in the A-Register.

Entry Points: ..MAP

Assembly: For 1 dimension:

```
CCA, <CLE>
LDB n (see below)
JSB ..MAP
DEF base address
DEF 1st subscript
Return (address in A)
```

For 2 dimensions:

```
CLA, <CLE>
LDB n (see below)
JSB ..MAP
DEF base address
DEF 1st subscript
DEF 2nd subscript
DEF length of 1st dimension
Return (address in A)
```

For 3 dimensions:

```
CLA, INA, <CLE>
LDB n (see below)
JSB ..MAP
DEF base address
DEF 1st subscript
DEF 2nd subscript
DEF 3rd subscript
DEF length of 1st dimension
DEF length of 2nd dimension
Return (address in A)
```

n = number of words per element in the array (1, 2, 3 or 4).

E-Register = 1 if store to this element.
0 if read from this element.

FORTRAN:	Not callable
Pascal:	Not callable
Result:	Integer
Errors:	None
Program TYPE:	Type 7, Utility
External References:	None
Notes:	This routine is available in 21MX FFP firmware. See Chapter 1.

/ATLG

Purpose:	Compute (1-x)/(1+x) in double precision.
Entry Points:	/ATLG
Assembly:	JSB /ATLG DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	None
Program TYPE:	Type 7, Utility
External References:	.TADD, .TSUB, .TDIV
Notes:	1. No error checking is performed. 2. The X- and Y-Registers may be changed.

/COS

Purpose:	.COS with no error return
Entry Points:	/COS
Assembly:	JSB /COS DEF *+3 DEF <result> DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	See .TSCS
Program TYPE:	Type 7, Utility
External References:	.COS, .ENTR

/CMRT

Purpose: Range reduction for SIN, .COS, .TAN, .EXP and .TANH.

Entry Points: /CMRT

Assembly:

```
LDA <flag>
JSB /CMRT
DEF <result>
DEF <constant>
DEF <argument>
Error return
Normal return      (B-Register contains least
                    significant bits of n)
```

FORTRAN: Not callable

Pascal: Not callable

Result: Double Real

Errors: See below for argument too large.

Program TYPE: Type 7, Utility

External References: .CFER, .TADD, .TSUB, .TMPY, .TFXD, .TFTD, .FLUN, IFIX, FLOAT

Notes:

1. This routine may alter the X- and Y-Registers.
2. This routine should be used by system programs only.

/EXP

Purpose: .EXP with no error return.

Entry Points: /EXP

Assembly:

```
JSB /EXP
DEF *+3
DEF <result>
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Double Real

Errors: See .EXP

Program TYPE: Type 7, Utility

External References: .EXP

/EXTH

Purpose: Compute $2^n \times 2^z$ or TANH(z) for small double real z.

Entry Points: /EXTH

Assembly:
```
LDA <n>
JSB EXTH
DEF <result>
DEF <y>
Return
```

FORTRAN: Not callable

Pascal: Not callable

Result: Double Real

Errors: None

Program TYPE: Type 7, Utility

External References: .PWR2, .TADD, TRNL

Notes: 1. No error checking is performed. The final exponent will be in error by a multiple of 128 if overflow or underflow occurs.

/LOG

Purpose: .LOG with no error return.

Entry Points: /LOG

Assembly:
```
JSB /LOG
DEF *+3
DEF <result>
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Double Real

Errors: See .LOG

Program TYPE: Type 7, Utility

External References: .LOG, .ENTR

/LOGO

Purpose:	.LOGO with no error return.
Entry Points:	/LOGO or /LOGT
Assembly:	JSB /LOGO or /LOGT DEF *+3 DEF <result> DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	See .LOGO
Program TYPE:	Type 7, Utility
External References:	.LOGO, .ENTR

/SIN

Purpose:	Calculate the sine of double-real x with no error return.
Entry Points:	/SIN
Assembly:	JSB /SIN DEF *+3 DEF <result> DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	See .TSCS
Program TYPE:	Type 7, Utility
External References:	SIN, .ENTR

/SQRT

Purpose: .SQRT with no error return.

Entry Points: /SQRT

Assembly:

```
JSB /SQRT
DEF *+3
DEF <result>
DEF x
Return
```

FORTRAN 4X: Not callable
FORTRAN 77: Callable

Pascal: Callable

Result: Double Real

Errors: See .SQRT

Program TYPE: Type 7, Utility

External References: .SQRT, .ENTR

/TAN

Purpose:	.TAN with no error return.
Entry Points:	/TAN
Assembly:	JSB /TAN DEF *+3 DEF <result> DEF x Return
FORTRAN 4X:	Not callable
FORTRAN 77:	Callable
Pascal:	Callable
Result:	Double Real
Errors:	See .TAN
Program TYPE:	Type 7, Utility
External References:	.TAN, .ENTR

/TINT

Purpose:	Conversion of double precision to integer.
Entry Points:	/TINT
Assembly:	JSB /TINT DEF *+2 DEF <arguments> Return (result in A)
FORTRAN 4X:	Callable as IDINT with y option
FORTRAN 77:	Callable as IDINT with y option
Pascal:	Callable
Result:	Integer in A
Errors:	Overflow set if argument outside $[-2^{15}, 2^{15})$
Program TYPE:	Type 7, Utility
External References:	.TINT

Chapter 5
Library Subroutines

REIO

The REIO subroutine permits user programs to perform reentrant I/O and disc-resident programs to be swappable while performing I/O. REIO is a utility type library subroutine that is appended to each program that calls it (Type 7).

```
+------------------------------------------------------------+
|                                                            |
|  CALL REIO (ICODE, ICNWD, IBUFF, ILEN)                     |
|                                                            |
+------------------------------------------------------------+
|                                                            |
|  ICODE - Request code. 1 = read;  2 = write.               |
|                                                            |
|  ICNWD - Control word.  Specifies the  LU  (must be non-disc) |
|          involved  in the  I/O operation.  Can  also  specify |
|          driver-dependent information.                     |
|                                                            |
|  IBUFF - Data buffer.  Contains  the  data to be written in a |
|          write operation  or  data returned from a read opera- |
|          tion.                                             |
|                                                            |
|  ILEN  - Data length.  Positive  number of words (or negative |
|          number of characters) to be read or written.      |
|                                                            |
+------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

ICODE, IBUFF, and ILEN are identical to those used in the EXEC 1 and EXEC 2 calls. ICNWD is similiar except that the Z-bit is not used (no control buffer passage) and the LU cannot specify a disc device.

REIO will always perform the requested I/O operation. However, it will perform the operation on a reentrant basis only if the buffer is less than 130 words (to save System Available Memory), and the buffer address is at least three words beyond the beginning of the program. Note that FORTRAN puts equivalenced arrays at the beginning of the program. Therefore, a dummy array (of at least three words) should be equivalenced prior to equivalencing the buffer to be used in the REIO call.

Error processing and the returns for the A- and B-Registers are the same as for the EXEC 1 and 2 calls.

BINRY

The BINRY subroutine is called from a FORTRAN program to transfer information to or from a disc device. BINRY has two entry points; BREAD for read operations and BWRIT for write operations.

```
+--------------------------------------------------------------------+
|                                                                    |
|  CALL BWRIT (IBUFF, ILEN, IDISC, ITRAK, ISECT, IOFST)              |
|                                                                    |
|  CALL BREAD (IBUFF, ILEN, IDISC, ITRAK, ISECT, IOFST)              |
|              -----                                                 |
+--------------------------------------------------------------------+
|                                                                    |
|  IBUFF - Data buffer.  Contains data to be written for a write     |
|          operation  or  data returned  from  a  read operation     |
|          (must be a non-EMA buffer).                               |
|                                                                    |
|  ILEN  - Data length.   The  number  of  words  to  be read or     |
|          written.                                                  |
|                                                                    |
|  IDISC - Disc LU.  Logical unit number of disc device involved     |
|          in the data transfer.                                     |
|                                                                    |
|  ITRAK - Disc track number.                                        |
|                                                                    |
|  ISECT - Disc sector number.                                       |
|                                                                    |
|  IOFST - Sector offset.   Offset (in words) within the sector.     |
|                                                                    |
|          If IOFST = 0,  the transfer starts  at  the  sector       |
|                         boundary.                                  |
|                                                                    |
|          If IOFST = n,  the transfer  starts n words past the      |
|                         beginning of the sector.                   |
|                                                                    |
+--------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

Since data transfer between a user's program and a disc device are buffered through the driver module on a sector basis, certain sector offset considerations must be allowed. These sector offset considerations are summarized below:

1. Offset=n (transfer begins within a sector), and less than a sector is written, or the data transfer ends on a sector boundary. The entire first sector is initially read into the driver's internal buffer, the data is modified according the BWRIT statement, and the entire sector is then rewritten on the disc with no data loss. No special precautions are required in this instance.

2. Offset=0 (transfer begins on a sector boundary), and less than a sector is written. The remaining data in the sector will be lost unless the entire existing sector on the disc is first read into a user's buffer, modified to reflect the desired changes, and then rewritten on the disc as a full sector.

3. Offset=0 or n, and a sector boundary is crossed in the data transfer. The remaining data in the final sector will be lost unless the entire final sector (of the data transfer) on the disc is read into a user's buffer, modified to reflect the desired changes, and then rewritten on the disc as a full sector.

It is the user's responsibility to initiate the "read-before-write" operation in case 2 and 3 above, the system automatically handles case 1.

RNRQ

The RNRQ subroutine allows cooperating programs a method of efficiently utilizing resources through a resource numbering scheme.

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL RNRQ (ICON, IRN, ISTAT)                                    |
|                        -----                                     |
+------------------------------------------------------------------+
|                                                                  |
|  ICON  - Control option. Defines how the resource number is to  |
|          be used.                                                |
|                                                                  |
|  IRN   - Resource number.                                        |
|                                                                  |
|  ISTAT - Status word. The status word returns are as follows:   |
|                                                                  |
|          0 - normal deallocate return                            |
|          1 - RN is clear (unlocked)                              |
|          2 - RN is locked locally to caller                      |
|          3 - RN is locked globally                               |
|          4 - no RN available now                                 |
|          6 - RN locked locally to other program                  |
|          7 - RN was locked globally when request was made        |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

A resource number is used when one program wishes to use a resource exclusively with the cooperation of other programs in the system. This resource could be a physical device or the system itself.

All programs must agree that a certain RN will be used as a lock or busy indicator for a given device.

Figure 5-1 illustrates the format of the control word required in the calling sequence.

15	14	5	4	3	2	1	0
WAIT OPTION		ALLOCATE OPTION			SET OPTION		
NO WAIT	NO ABORT	CLEAR	GLOBAL	LOCAL	CLEAR	GLOBAL	LOCAL

Figure 5-1. Control Word Format (ICON)

If more than one bit is set in the control word, the following order of execution is used:

1. local allocate (skip step 2 if done)
2. global allocate
3. deallocate (exit if done)
4. local set (skip step 5 if done)
5. global set
6. clear

The system has a set quantity of resource numbers (RNs) that are specified during generation. If a resource number is not available when a program requests one, the program is suspended until one is free, unless the "no-wait" bit is set. If the "no-wait" bit is set, the RN location is set to zero. If the RN allocation is successful, the value returned in IRN is set by the system. It has no meaning to the user but must be specified (through IRN) when a lock is requested or the RN is cleared or deallocated.

The "no-abort" bit is used to alter the error return point of the call as shown in the following example:

```
                                         .
                                         .
                              ICON = ICON + 40000B
                              CALL RNRQ (ICON,...)
  error routine        --->   GOTO 100
  normal return point ---->          .
                                     .
                                     .
                              100 error processing
                                     .
```

The above special error return is established by setting bit 14 to 1 in the request code word (ICON). This causes the system to execute the GOTO statement following CALL RNRQ if there is an error, or skip the GOTO statement if there is no error. If the error return is taken, error information will be available in the A- and B-Registers.

RNRQ ALLOCATE OPTIONS:

LOCAL - Allocate an RN to the calling program. The number is returned in the IRN parameter. The number is automatically released on termination of the calling program, and only the calling program can deallocate the number.

GLOBAL - Allocate an RN globally. The number is released by a request from any program.

CLEAR - Deallocate the specified number.

RNRQ SET OPTIONS:

LOCAL - Lock the specified RN to the calling program. The RN is specified in the IRN parameter. The local lock is automatically released on termination of the calling program. Only the calling program can clear the number.

GLOBAL - Lock the specified RN globally. The RN is specified in the IRN parameter and the calling program can globally lock this number more than once. The number is released by a request from any program.

CLEAR - Unlock the specified RN.

If the RN is already locked to someone else, the calling program is suspended (unless the "no-wait" bit is set) until the RN is cleared. If more than one program is attempting to lock an RN, the program with the highest priority is given precedence. A single call can both lock and clear an RN.

If a program makes this call with the clear bit set, in addition to either the global or local set bits, the program will wait (in the general wait list) until the RN is cleared by another program and then continue with the RN clear.

An entry point is provided for drivers or privileged subroutines of Type 3 programs that wish to clear a global (and only global) RN:

```
LDA     RN
JSB     $CGRN
return  point
```

LURQ

The LURQ subroutine allows a program to exclusively dominate (lock) an input/output device.

```
+------------------------------------------------------------------+
|                                                                  |
| CALL LURQ (ICON, LURAY, NUM)                                     |
|                                                                  |
+------------------------------------------------------------------+
|                                                                  |
| ICON  - Control option.  An  octal  number  that specifies the  |
|         locking or unlocking action to be performed.            |
|                                                                  |
| LURAY - LU array.   An  array  of  LU numbers  to be locked or  |
|         unlocked.                                                |
|                                                                  |
| NUM   - The number of LUs to be locked or unlocked.             |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

This request temporarily assigns a logical unit to the calling program. It prevents a higher priority program from interrupting a program's use of the device until the device is unlocked by the program that locked it.

The LURQ routine request allows up to 31 programs to exclusively dominate (lock) input/output devices. Any other program attempting to use or lock a locked LU will be suspended until the original program unlocks the LU or terminates.

The functions of the control option (ICON) are summarized below:

* ICON = 000000B - unlock LUs specified in LURAY.

* ICON = 100000B - unlock all LUs the program currently has locked.

* ICON = 000001B - lock (with wait) the specified LUs.

* ICON = 100001B - lock (without wait) the specified LUs.

NO-ABORT BIT (bit 14) - The "no-abort bit" is used to alter the error return point of this call as shown in the following example:

```
                                              .
                                         ICON = ICON + 40000B
                                         CALL LURQ (ICON,..)
     error routine          ----->       GOTO 100
     normal return point ------>              .
                                              .
                                              .
                                          100 error processing
                                              .
```

The above special error return is established by setting bit 14 in ICON (ICON = ICON + 40000B). This causes the system to execute the GOTO statement following the CALL LURQ if there is an error, or to skip the GOTO statement if there is no error. If the error return is taken, error information will be available in the A- and B-Register.

DISC ALSO BIT (bit 11)---The "disc also" bit is used to allow LU locks on discs. Failure to set this bit when specifyng a disc LU lock request will result in LU 2 abort error.

UNLOCK---To unlock all owned LUs, the LURAY array is not used but still must be coded; the program will not abort.

Any LUs the program has locked will be unlocked when the program:

* Performs a standard termination.

* Performs a serial reusability termination.

* Aborts.

The LUs will not be unlocked when the program performs a "save resources" termination.

This subroutine calls the program management subroutine (RNRQ) for a resource number (RN) allocation; that is, the system locks an RN locally to the calling program. Therefore, before the logical unit lock subroutine can be used, a resource number must have been defined during generation. Only the first 31 RNs can be used for LU locks.

If the "no-wait" option is coded, the A-Register will contain the following information on return:

- 0 - LU lock successful
- -1 - no RN available at this time
- 1 - one or more of the LUs is already locked to some other program.

The calling program may not have LUs locked at the time of the call unless the no-wait option is used. All LUs locked by the calling program are locked to the same RN.

PARSE ($PARS)

The PARSE subroutine (or $PARS) allows a program to parse a string into separate parameters.

```
+-------------------------------------------------------------+
|                                                             |
|  CALL PARSE (IBUFA,ICON,IRBUF)                              |
|                         -----                               |
+-------------------------------------------------------------+
|                                                             |
|  IBUFA - Source buffer. Contains the ASCII string to be parsed.|
|                                                             |
|  ICON  - Character count.  Number of characters  in the ASCII |
|          string.                                            |
|                                                             |
|  IRBUF - Receiving buffer.  A 33-word buffer that contains the |
|          results of the parse operation.                    |
|                                                             |
+-------------------------------------------------------------+
```

FORTRAN: Not callable

Pascal: Not callable

The results of the parse operation are stored in IRBUF using four words of IRBUF to represent each parameter found in IBUFA. The function of these four words is summarized below:

WORD	ENTRY	
1	FLAG WORD	0 = NULL 1 = NUMERIC 2 = ASCII
2	VALUE(1)	0 If NULL; Value if Numeric; first two characters if ASCII.
3	VALUE(2)	0 If Null or numeric else the 3rd and 4th characters (ASCII).
4	VALUE(3)	0 If NULL or numeric else the 5th and 6th characters (ASCII).

ASCII parameters are separated from numeric parameters by examination of each character. One or more non-digit characters (except a trailing "B or b" or leading "-") makes a parameter ASCII. This subroutine can parse up to eight parameters.

IRBUF is initialized to 0 by the system before parsing the string contained in IBUFA.

Word 33 of IRBUF will be set to the number of parameters in the string.

The PARSE routine ignores all blanks and expects commas to delimit the parameters. ASCII parameters are padded to six characters with blanks or if more than 6 characters, the left-most 6 are used. Numbers may be negative (leading "-") and/or octal (trailing "B or lower case b").

The format for the Assembly language version of the PARSE subroutine is as follows:

```
 .
 .
EXT     $PARS
 .
 .
LDA     IBUFA
LDB     ICON
JSB     $PARS
DEF     IRBUF
 .
 .
```

INPRS

The INPRS subroutine converts a buffer of parsed data (produced by the PARSE subroutine) back to its original form.

```
+-------------------------------------------------------------------+
|                                                                   |
|  CALL INPRS (IRBUF, NUPAR)                                        |
|                                                                   |
+-------------------------------------------------------------------+
|                                                                   |
|  IRBUF - Parameter buffer.  Contains the parameters to be con-    |
|          verted back to an ASCII string; previously produced by   |
|          the PARSE routine.                                       |
|                                                                   |
|  NUPAR - Number of parameters contained in IRBUF; obtained from   |
|          word 33 of IRBUF returned by PARSE (NUPAR = IRBUF(33)).  |
|                                                                   |
+-------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

IRBUF and NUPAR are obtained from a previous call to the PARSE routine or are formatted by the user as if they were; the function of INPRS is to reverse the action performed by PARSE.

The results of the INPRS operation are stored in IRBUF. The length of the resultant ASCII string will be eight times the number of parameters (8 X NUPAR).

$CVT3 (CNUMD,CNUMO), $CVT1 (KCVT)

These subroutines convert a positive integer binary number to ASCII.

```
+------------------------------------------------------------------+
|  CALL CNUMD ( n ,IBUF)                                           |
|  CALL CNUMO ( n ,IBUF)                                           |
|  I= KCVT (n)                                                     |
+------------------------------------------------------------------+
|  n    - Actual binary number that is to be converted to ASCII;  |
|         must be positive.                                        |
|                                                                  |
|  IBUF - Three-word array that the ASCII representation (6       |
|         characters) of n is returned to.  For CNUMD, a decimal  |
|         representation is returned; for CNUMO, an octal repre-  |
|         sentation is returned.  Leading zeros are suppressed    |
|                                                                  |
|  I    - Least significant two digits of the ASCII decimal       |
|         representation of n.                                     |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembler Language formats for these routines are shown below:

```
        :
       EXT        $CVT3 (or $CVT1)
        :
       LDA        n
        .
       CLE             (for octal results; E=0)
       or
       CCE             (for decimal results; E=1)
        .
       JSB        $CVT3 (or $CVT1)
       return
        :
```

Upon return the register contents will be as follows:

E-Register = 1

A-Register = $CVT3 - address of 3-word ASCII result.
             $CVT1 - two least-significant characters of converted number.

B-Register = unchanged.

MESSS

The MESSS subroutine provides programmatic access (limited by the user capability if under session control) to all system commands.

```
+--------------------------------------------------------------------+
|                                                                    |
|  IA = MESSS (IBUF,INUM [,LU])                                      |
|                                                                    |
+--------------------------------------------------------------------+
|                                                                    |
|  IA   - 0 if no message is returned from the system;  negative     |
|         character count if message is returned (same as return     |
|         of A-Register).                                            |
|                                                                    |
|  IBUF - Command buffer. Contains the ASCII command to be pass-      |
|         ed to the RTE-6/VM Operating System.                       |
|                                                                    |
|  INUM - Character count. Number of characters (bytes) contain-     |
|         ed in the ASCII command.                                   |
|                                                                    |
|  LU   - Replacement LU.  If the command in IBUF is a RU or an      |
|         ON, and the first parameter in the parameter string is     |
|         zero or absent,  then LU will be inserted as the first     |
|         parameter.                                                 |
|                                                                    |
+--------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

If the operating system returns information, it will be placed in IBUF and the character count (negative) is placed in IA (and the A-Register). The command buffer (IBUF) should be at least 14 words long to allow for the largest possible system return.

If the command was a RU or ON command, the father ID-segment-word 33 will be propagated to the son's ID-segment-word 33.

The Assembly Language format of the MESSS routine is as follows:

```
             .
             .
             JSB  MESSS
             DEF  RTN
             DEF  IBUF ---+
             DEF  INUM    |-parameter addresses
             DEF  LU   ---+
       RTN   .
             .
             .
       IBUF  BSS   ---+
       INUM  DEC      |-parameter values
       LU    DEC   ---+
```

On return the A-Register will be 0 if no information is returned, or will be the negative character count if information is returned.

COR.A, COR.B

The COR.A subroutine returns the address of the first word of available memory (high address +1) for a main program or program segment given the address of the main program or segment associated with the ID-segment specified.

The COR.B subroutine returns the first word of available memory for a main program given the address of its ID segment. For segmented programs, the address returned is equal to:

main high address + largest segment high address +1.

Assembly Language calling sequence:

```
        :                      :
        EXT   COR.A            EXT   COR.B
        :                      :
        LDA   IDSEG            LDA   IDSEG
        JSB   COR.A            JSB   COR.B
        -return                -return-
        :                      :
```

IDSEG - ID segment address; must be long ID segment address for COR.B.

FORTRAN: Not callable

Pascal: Not callable

For non-segmented programs, the addresses returned by COR.A and COR.B are the same.

On return from COR.A, the A-Register contains the high address +1 of the main program or segment associated with the ID segment specified. The B-Register is not used.

On successful returns from COR.B, the B-Register will contain the high address +1 of the main program associated with the specified ID segment. If the program is segmented, the B-Register will contain the high address +1 of the largest segment. The A-Register will contain 0.

On unsuccessful returns from COR.B, the A-Register will contain -1, and the B-Register will be meaningless. COR.B makes an error return if it is passed the address of a short ID segment.

.DRCT

The .DRCT subroutine resolves an indirect address within the map of the calling program.

Assembly Language calling sequence:

```
EXT   .DRCT
 .
 .
JSB   .DRCT
DEF   ADDR
-return-
```

ADDR - Address to be resolved.

FORTRAN 77: Callable

Pascal: Callable

The .DRCT subroutine returns with the A-Register set to the direct address of ADDR, the B-Register unaltered, and the E-Register lost. This routine is usually used when ADDR is external.

FTIME

The FTIME subroutine returns the date and time as an ASCII string.

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL FTIME (IBUF)                                               |
|              ----                                                |
|                                                                  |
+------------------------------------------------------------------+
|                                                                  |
|  IBUF - 15-word array that receives the ASCII string.            |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The format of the returned string is illustrated by the following example:

12:42 PM FRI., 23 OCT., 1981

The month will be returned as a three-character abbreviation followed by a period or (June, July) as four characters.

GETST

The GETST subroutine recovers the parameter string from a program's command string storage area. The parameter string is defined as all the characters following the second comma in the command string (third comma if the first two characters in the first parameter are NO).

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL GETST (IBUF, ILEN, ILOG)                                   |
|              ----        ----                                    |
+------------------------------------------------------------------+
|                                                                  |
|  IBUF - String buffer.  Array to which the parameter string is  |
|         returned.                                                |
|                                                                  |
|  ILEN - String length. Requested number of words (if positive)  |
|         or characters (if negative) to be returned.              |
|                                                                  |
|  ILOG - Transmission log. Actual number of words or characters  |
|         returned.                                                |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembly Language calling sequence for the GETST subroutine is as follows:

```
          EXT  GETST
           .
           .
          JSB  GETST
          DEF  RTN
          DEF  IBUF
          DEF  ILEN
          DEF  ILOG
    RTN    .
           .
           .
    IBUF  BSS  n
    ILEN  DEC  n
    ILOG  NOP
           .
```

Upon return, ILOG contains a positive integer giving the number of words (or characters) transmitted. The A- and B-Registers may be modified by GETST. Note that if RMPAR is used, it must be called before GETST.

When an odd number of characters is specified, an extra space is transmitted in the lower byte of the last word.

An EXEC 14 call can be used to recover the entire command string (including the parameter string).

PRTN, PRTM

The PRTN and PRTM subroutines are used by the calling program to pass parameters back to its father (the program that scheduled it). The father can recover the returned parameters by calling the RMPAR routine.

```
+----------------------------------------------------------------+
|                                                                |
|   CALL PRTN (IPRAM)                                            |
|                                                                |
|   CALL PRTM (IPRAM)                                            |
|                                                                |
+----------------------------------------------------------------+
|                                                                |
|   IPRAM - Parameter buffer.  A 5-word array (for PRTN) or a    |
|           4-word array (for PRTM) that contains the parameters |
|           to be returned to the father program.                |
|                                                                |
+----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The PRTN routine passes five parameters and clears the WAIT FLAG. Since the WAIT FLAG is cleared, the calling program should terminate immediately after the call. For example;

```
DIMENSION IPRAM (5)
   .
   .
CALL PRTN (IPRAM)
CALL EXEC (6)
```

The PRTM routine passes four parameters and does not clear the WAIT FLAG; an immediate termination (EXEC 6) is not necessary. When the parameters are recovered with RMPAR, the first parameter will be meaningless.

The Assembly Language calling sequences for the PRTN and PRTM routines is shown below:

```
        EXT   EXEC, PRTN
         .
         .
        place values in IPRAM
         .
        JSB   PRTN
        DEF   *+2
        DEF   IPRAM
        JSB   EXEC
        DEF   *+2
        DEF   SIX
         .
         .
IPRAM   BSS   5
SIX     DEC   6


        EXT   PRTM
         .
         .
        place values in IPRAM
         .
        JSB   PRTM
        DEF   *+2
        DEF   IPRAM
         .
         .
IPRAM   BSS   4
```

IFBRK

The IFBRK subroutine tests the BREAK FLAG and clears it if it is set. The BREAK FLAG is set with the BR command described in the RTE-6/VM Terminal User's Reference Manual.

```
+----------------------------------------------------------------+
|                                                                |
|   IF (IFBRK (IDMY)) n, m                                       |
|                                                                |
+----------------------------------------------------------------+
|                                                                |
|   IDMY - A dummy variable  used  to inform the FORTRAN compiler |
|          that an external function is being called.            |
|                                                                |
|   n    - Statement number that control branches to if the break |
|          flag was set; the flag will be cleared.               |
|                                                                |
|   m    - Statement number that control branches to if the break |
|          flag is not set.                                      |
|                                                                |
+----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembly Language calling sequence for the IFBRK routine is as follows:

```
 .
EXT  IFBRK
 .
 .
JSB  IFBRK
DEF  *+1
 .
 .
```

On return, the A-Register will contain -1 if the BREAK FLAG was set or will contain 0 if it was not set. The flag will be cleared if it was set.

IDGET

The IDGET subroutine retrieves the ID Segment address of a specified program.

```
+--------------------------------------------------------------+
|                                                              |
|   IDSEG = IDGET (INAM)                                       |
|   -----                                                      |
+--------------------------------------------------------------+
|   IDSEG - ID segment address.  Contains the returned ID segment |
|           address  of  the specified program;  set to 0  if the |
|           program does not exist.                               |
|                                                                 |
|   INAM  - Program name.   3-word array  used to contain  the 5- |
|           character ASCII name of  the  program to which the ID |
|           segment address is being requested.                   |
+--------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembly Language calling sequence for the IDGET routine is as follows:

```
           :
          EXT     IDGET
           :
          JSB     IDGET
          DEF     *+2
          DEF     INAM
           :
    INAM  ASC 3,PROGY
           .
```

On return, the following registers are set as indicated:

A-Register = ID segment address, or 0 if not found

E-Register = 0 if program found, or 1 if not found

B-Register = 0

TMVAL

The TMVAL subroutine reformats the system millisecond time format (double-word negative integer) into an array of time parameters.

```
+--------------------------------------------------------------+
|                                                              |
|  CALL TMVAL (ITIM, ITMR)                                     |
|                    ----                                      |
+--------------------------------------------------------------+
|  ITIM - Two-word negative time value in tens of milliseconds.|
|         This double-word integer can be obtained from the    |
|         system entry point $TIME (real-time clock) or the time|
|         value stored in a program's ID segment.              |
|                                                              |
|  ITMR - Time array.  5-word array to which the system returns|
|         the reformatted time.  The array is set up as:       |
|                                                              |
|         ITMR(1) = tens of milliseconds                       |
|         ITMR(2) = seconds                                    |
|         ITMR(3) = minutes                                    |
|         ITMR(4) = hours                                      |
|         ITMR(5) = day of year (julian)  -  not related to call|
|                   values.                                    |
+--------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The next scheduled execution time of a program currently in the time list can be obtained from the program's ID segment and then formatted into an array of time parameters by the TMVAL routine.

```
      DIMENSION   INAM(3),ITMR(5),IARRAY(2)
      DATA INAM/2HPR,2H06,2H1/
        :
      IDSEG=IDGET (INAM)
      ITAD=IDSEG+18
        :
      IARRAY(1)=IGET(ITAD)
      IARRAY(2)=IGET(ITAD+1)
        :
      CALL TMVAL (IARRAY,ITMR)
```

EQLU

The EQLU subroutine finds the logical unit number of an interrupting device if given the address of word 4 of the device's Equipment Table entry.

```
+-----------------------------------------------------------------+
|                                                                 |
|  CALL EQLU (LU)                                                 |
|             --                                                  |
+-----------------------------------------------------------------+
|                                                                 |
|  LU - Logical  unit  number  of  interrupting device  (same as  |
|       A-Register return).                                       |
|                                                                 |
+-----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The EQLU routine expects the address of EQT word 4 of the interrupting device to be in the B-Register. This is done by another program/subroutine (using LDB EQT4) or by the driver associated with the interrupting device.

The EQLU routine will function correctly only if the LU number to be returned is less than or equal to 99.

The Assembly Language format is as follows:

```
          EXT  EQLU
           :
          JSB  EQLU
          DEF  RTN
          DEF  LU
     RTN   :
```

On return:

A-Register = 0 if an LU referring to the EQT was not found.
= LU if LU was found.

B-Register = ASCII "00" (if LU not found) or the ASCII LU number.

LU = Same as A-Register.

TRMLU

The TRMLU subroutine finds the logical unit number of an interrupting device if given the address of word 4 of its Equipment Table entry, and checks that it is an interactive device.

```
+-----------------------------------------------------------------+
|                                                                 |
|  CALL TRMLU (LU)                                                |
|              --                                                 |
+-----------------------------------------------------------------+
|                                                                 |
|  LU - Logical unit  number  of  interrupting device.  (Same as  |
|       A-Register return).                                       |
|                                                                 |
+-----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The TRMLU routine expects the address of EQT word 4 of the interrupting device to be in the A-Register. This is done by another program/subroutine (using LDA EQT4) or by the driver associated with the interrupting device.

The TRMLU routine will function correctly only if the LU number to be returned is less than or equal to 99.

The Assembly Language format is as follows:

```
          EXT  TRMLU
           :
          JSB  TRMLU
          DEF  RTN
          DEF  LU
     RTN   :
```

On return:

```
A-Register = 0 if LU not found.
           = LU number if found.

B-Register = ASCII "00" if LU not found, ASCII LU number if found.

LU         = Same as A-Register. (optional).
```

IFTTY

The IFTTY subroutine determines whether a logical unit is interactive or not.

```
+----------------------------------------------------------------+
|                                                                |
|  INT = IFTTY (LU)                                              |
|  ---                                                           |
+----------------------------------------------------------------+
|                                                                |
|   INT - Set to -1 if LU is interactive;  set to 0 if LU is not |
|         interactive  (same as A-Register return).              |
|                                                                |
|   LU  - Logical unit number of device being tested.            |
|                                                                |
+----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembly Language calling sequence for the IFTTY routine is as follows:

```
         EXT  IFTTY
          .
          .
         JSB  IFTTY
         DEF  RTN
         DEF  LU
    RTN   .
          .
    LU   DEC  n      Logical unit being tested.
```

On return, the following registers are set as indicated:

A-Register = -1 if LU is interactive, 0 if it is not interactive.

B-Register = Upper byte is the driver type (word 5 of Equipment Table entry, bits 8-13). Lower byte is the subchannel number.

LOGLU

The LOGLU subroutine returns the logical unit numbers of the terminal from which the currently executing program was scheduled.

```
+------------------------------------------------------------------+
|                                                                  |
|   LU = LOGLU (LUSYS)                                             |
|   --                                                             |
+------------------------------------------------------------------+
|                                                                  |
|   LU    - Logical unit number  of device from which the calling  |
|           program was scheduled  (same as A-Register); 1 (if in  |
|           session) or positive log-LU (if not in session).       |
|                                                                  |
|   LUSYS - The system LU of the session terminal (in session) or  |
|           the negative log-LU (not in session).                  |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The LOGLU routine will return the LU numbers of the console from which the currently executing program was scheduled. This LU number is passed down from the Father program to the Son program when one program schedules another program for execution. If the program was scheduled by interrupt or from the time list, the scheduling LU will be LU 1, the system console.

The Assembly Language calling sequence is as follows:

```
          EXT  LOGLU
           .
           .
          JSB  LOGLU
          DEF  RTN
          DEF  LUSYS
     RTN   .
           .
```

On return:

A-Register = LU number of device from which program was scheduled.

B-Register = ASCII LU number.

LUTRU

The LUTRU subroutine returns the true system logical unit number associated with a session or batch LU.

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL LUTRU (ITEST, LU)                                          |
|                     --                                           |
|     or                                                           |
|                                                                  |
|  LU = LUTRU (ITEST)                                              |
|  --                                                              |
+------------------------------------------------------------------+
|                                                                  |
|  LU    - True system LU is returned here.                        |
|                                                                  |
|  ITEST - The session logical unit number to be checked.          |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

If the calling program is not a session or batch program, LU is set equal to ITEST.

If the calling program is a session program and ITEST is not defined for the caller's session, LU is set to -1.

If the calling program is a batch program and ITEST is not defined in the Batch Switch Table, LU is set equal to ITEST.

LUSES

The LUSES subroutine scans the list of Session Control Blocks (SCBs) looking for a SCB defined for the session identifier passed in the call.

```
+-------------------------------------------------------------+
|                                                             |
|  ISCB = LUSES (IDBNT)                                       |
|  ----                                                       |
+-------------------------------------------------------------+
|                                                             |
|  ISCB  = Address of SST length word in SCB if found, 0 if not|
|          found (same as A-Register return).                 |
|                                                             |
|  IDBNT = Session identifier (word 3 of SCB).                |
|                                                             |
+-------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The Assembly Language format is as follows:

```
         EXT  LUSES
          .
          .
         JSB  LUSES
         DEF  RTN
         DEF  IDENT
     RTN  .
          .
```

On return:

A-Register = 0 if SCB not found.

= SCB address if found.

GTERR

The GTERR subroutine returns the SCB error mnemonic from the current Session Control Block (SCB).

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL GTERR (INMIC [,IERR])                                      |
|              -----   ----                                        |
|                                                                  |
+------------------------------------------------------------------+
|                                                                  |
|  INMIC - 4-word buffer that error mnemonic is returned to.       |
|                                                                  |
|  IERR  - Error return.  0 indicates that the retrieval was       |
|          successful; -1 indicates that the calling program was   |
|          not in session (optional).                              |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The error mnemonic is an 8-ASCII-character message that represents the last error encounted for the current session. The error mnemonic is posted by the RTE-6/VM Operating System and some HP supported subsystems. The error mnemonic can also be updated by a user application program by calling the library PTERR routine.

Note the HELP command (see RTE-6/VM Terminal User's Reference Manual) uses the 8-character mnemonic as the implicit keyword to search the HELP file and returns expanded information on the last error posted in the current session.

The error mnemonic is stored in words 5 through 8 of the SCB.

PTERR

The PTERR subroutine updates the error mnemonic in the current Session Control Block (SCB).

```
+------------------------------------------------------------------+
|                                                                  |
|  CALL PTERR (INMIC [,IERR])                                      |
|                    ----                                          |
+------------------------------------------------------------------+
|                                                                  |
|  INMIC - 4-word buffer  that contains the error mnemonic to be   |
|          posted to the SCB.                                      |
|                                                                  |
|  IERR  - Error return.  0 indicates a successful posting oper-   |
|          ation;  -1 indicates that the calling program was not   |
|          in session (optional).                                  |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

The PTERR routine places an 8-ASCII-character (4-words) message in the current SCB. This message can be retrieved by calling the library GTERR routine. The PTERR/GTERR calls can be used to implement error and data communication schemes between programs running in the current session.

SESSN

The SESSN subroutine determines if the calling program is in session.

Assembly Language calling sequence:

```
              .
              .
              JSB   SESSN
              DEF   RTN
              DEF   ID
        RTN   -return-
```

ID - ID segment address of program that is to be checked.

FORTRAN 77: Callable

Pascal: Callable

On return from SESSN:

E-Register = 0 if calling program was in session.
1 if calling program was not in session.

B-Register = ID segment session word (address of SST length word in SCB) if calling program was in session.

The ID segment address of the calling program can be obtained by using the library IDGET routine.

ICAPS

The ICAPS subroutine returns the current session's capability level.

```
+------------------------------------------------------------------+
|                                                                  |
|   ICPSE = ICAPS (IDUMY)                                          |
|                                                                  |
+------------------------------------------------------------------+
|                                                                  |
|   IDUMY - Dummy variable.                                        |
|                                                                  |
|   ICPSE - Session capability;  0  if calling program was not in  |
|           session.                                               |
|                                                                  |
+------------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

SYCON

The SYCON subroutine writes a message to the system console (system LU 1).

```
+----------------------------------------------------------------+
|                                                                |
|  CALL SYCON (IBUF, ILEN)                                       |
|                                                                |
+----------------------------------------------------------------+
|                                                                |
|  IBUF - Buffer that contains the message to be written.        |
|                                                                |
|  ILEN - Length of IBUF, positive value indicates the number of |
|         words and a negative value indicates the number of     |
|         characters.                                            |
|                                                                |
+----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

This routine bypasses the Session Switch Table (SST) and writes directly to system LU 1.

SEGLD

SEGLD loads a segment (background or real-time) of the calling program from disc into an overlay area in memory provided by the program, and transfers control to the segment's entry point.

```
CALL SEGLD(INAM,IERR[,PRAM1[,PRAM2[,PRAM3[,PRAM4[,PRAM5]]]]])
```

NAME is a three-word array containing the five-character segment name:

NAME(1) = 1st two characters
NAME(2) = 2nd two characters
NAME(3) = last character in upper 8 bits (the lower byte is not significant)

IERR is an error return; its value is -6 if the segment cannot be loaded.

PRAM1 through PRAM5 are up to five optional parameters that the segment may recover via RMPAR.

SEGLD loads segments via an EXEC 8 request. Control returns to the calling routine if the segment cannot be loaded (IERR will be equal to -6) or if the segment calls SEGRT (IERR will equal zero). The advantage of SEGLD over EXEC 8 is that segments can return to the main program if SEGLD is used; EXEC 8 does NOT allow a return to the main as its standard operation.

PRAM1 through PRAM5 can be passed to the segment to be loaded. The segment should use RMPAR to retrieve these parameters.

GTSCB

The GTSCB subroutine returns the contents of the current SCB.

```
+---------------------------------------------------------------+
|                                                               |
|  CALL GTSCB (IBUF,ILEN,IERR[,ADSCB])                          |
|              ----      ----                                   |
|                                                               |
|---------------------------------------------------------------|
|                                                               |
|  IBUF -  Buffer to which the contents of the SCB are returned,|
|          beginning with word 3.                               |
|                                                               |
|  ILEN -  Length of IBUF.                                      |
|                                                               |
|  IERR -  If positive, indicates the number of words in the SCB|
|          (not counting words 0, 1, and 2).   If -1, indicates |
|          the calling program was not in session. If -2,       |
|          indicates the SCB address passed to GTSCB was not a  |
|          valid address.  If a negative number (besides -1 or  |
|          -2), indicates that the buffer size passed to GTSCB  |
|          (IBUF) was too small;  the negative number indicates |
|          the actual SCB size (not counting words 0, 1, and 2).|
|                                                               |
|  ADSCB - Address of SCB (optional).                           |
|                                                               |
+---------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

LIMEM

LIMEM allows a program to use and manage the memory between the end of its code and the end of memory available to the program. LIMEM finds and returns the limits of available memory.

For non-segmented programs:

```
CALL LIMEM (CODE [,FWAM [,WORDS]])
```

CODE	is positive or zero to lock memory, or negative to release memory.
FWAM	is the returned address of the first word of available memory after the program code.
WORDS	is the returned number of words of memory available after FWAM.

For segmented programs:

```
CALL LIMEM (CODE [,FWAM [,WORDS [,CURNT [,CWRDS]]]])
```

CODE	if negative, releases memory and returns to original FWAM. If positive or zero, CODE gets memory.
FWAM	is the returned first word of available memory after the largest segment (or after the main if the program is not segmented).
WORDS	is the returned number of words available after the largest segment (or after the main if the program is not segmented).
CURNT	is the returned first word available after the current segment (or after the main if no segments have been loaded).
CWRDS	is the returned number of words available after the current segment (or after the main if no segments have been loaded).

CLRQ

The CLRQ subroutine allows the assignment of ownership to a class number so that in the event of a program terminating or aborting without cleaning up the classes and class buffers assigned to it, the system will be able to deallocate these resources

This routine also allows programmatic flushing of pending class buffers on an LU or flushing of all class buffers (pending or completed) with deallocation of the class resource itself.

```
+-----------------------------------------------------------------+
|                                                                 |
|   CALL CLRQ (FUNC,CLASS[,PRAM1])                                |
|                                                                 |
|-----------------------------------------------------------------|
|                                                                 |
|  FUNC  - Class management control function.   Values are 1, 2, |
|          and 3.                                                 |
|                                                                 |
|             15 14 13 12 11 10  9  8  7  6  5  4  3  2  1  0    |
|            +-----------------------------------------------+   |
|            |NW|NA|                                   | FUNC|   |
|            +-----------------------------------------------+   |
|                                                                 |
|  CLASS - Class number.                                          |
|                                                                 |
|  PRAM1 - Call  dependent parameter; used to describe a program |
|          name or LU.                                            |
|                                                                 |
+-----------------------------------------------------------------+
```

FORTRAN 77: Callable

Pascal: Callable

Class Management Parameters

The FUNC parameter has two bits that can be set to allow the user more control of his process and the ability to obtain error information from the registers.

Bit 15 is the "no-wait" bit. It allows the user not to be suspended if no class numbers are available when the CLRQ request is made. The A-Register will contain a -1 if no class numbers are available, otherwise the A-Register will contain 0 (zero) which means the request completed without error.

Bit 14 is the "no-abort" bit. It operates in the same fashion as the "no-abort" bit in the ICODE parameter of the EXEC calls. The setting of this bit allows the user to maintain program activity if an error is made in the call sequence of the CLRQ routine, or some other programming activity. When set, the registers will contain an ASCII error message. The A-Register will contain the first two ASCII characters and the B-Register will contain the second two ASCII characters of the four character message.

CL01 Illegal class number or no class numbers defined in system.

CL02 Parameter or calling sequence error.

FUNC=1 Class ownership is assigned. If PRAM1 contains the name of a program, the program is assigned ownership of the class specified in CLASS. If PRAM1 is zero, no ownership is assigned. If PRAM1 is defaulted, which in this case means omitting the parameter from the call, the calling program is assigned ownership. If CLASS is zero, a new class number is allocated by the call. When a program is the "owner" of a class, the system then knows when that class can be deallocated. When the program becomes dormant, then the system will deallocate the class and its associated buffers.

FUNC=2 Flush class requests and deallocate class specified in CLASS. All non-active pending requests will be deallocated. Abort requests will be issued by the system for all active I/O requests, in which case the buffer will be deallocated at the completion of abort processing. All previously completed requests will be immediately deallocated. The class table entry will be flagged so that no new requests will be issued on the class. An I/O error (IO00) will be returned to programs that issue a request on the class after the class table entry is flagged. When the pending class request count in the class table entry reaches zero, the system will deallocate the class.

NOTE: PRAM1 is not used.

FUNC=3 Flush class requests on LU designated by PRAM1. The system looks at the class table entry specified in CLASS. Non-active requests on CLASS that are pending on the LU specified in PRAM1 are deallocated. If a request is active, an abort request is issued by the system. The buffer will be deallocated when the active EXEC request is completed. The Class Number is not deallocated nor are the completed class buffers affected.

CLASS CLASS is the class number that can be owned by a program. The class number format is the same as in the EXEC 17, 18, 19 and 20 requests. This parameter works in conjunction with FUNC and PRAM1. For example, when this parameter is zero and FUNC=1 then a new class number will be assigned to the calling program and returned in CLASS when the call completes.

PRAM1 Is an optional parameter that works with FUNC and CLASS in a number of ways. See the above descriptions for details.

General Flow

The system will check all terminating and aborting programs for class ownership. If ownership exists, all completed class request buffers will be deallocated. If the program terminates without terminating its I/O requests, such as a program that does a CLASS READ and then terminates, the pending class requests will be allowed to be completed normally. If I/O is to be aborted, all non-active pending requests are flushed, and the drivers will be issued an abort request for all active requests. In the latter case, the buffer and the class will be automatically deallocated by the system when abort processing has completed.

The following example will allocate two class numbers, assigning one to the calling program, and the second to the program called P2. P2 must have an ID segment or an SC05 error will result. The "no-abort" bit is set in the function parameter to prevent the program from being aborted.

```
FTN4,L
      PROGRAM ALLOC
      IMPLICIT INTEGER (A-Z)
      DIMENSION PRAM1 (3)

      CLAS1 = 0
      FUNC = 1
C  ALLOCATE FIRST CLASS NUMBER TO THE CALLING PROGRAM
      CALL CLRQ (FUNC + 40000, CLAS1)

C  CHECK ERROR
      GOTO 100
C  NOW ALLOCATE THE 2ND CLASS NUMBER, ASSIGNING IT TO P2
   20 CLAS2 = 0
      PRAM1 = 2HP2
      CALL CLRQ (FUNC + 40000, CLAS2, PRAM1)

C  CHECK ERROR
      GOTO 100
        :
        :
  100 CONTINUE
C     ERROR PROCESSING ................................
        :
      END
```

LKEMA

The LKEMA subroutine locks a shareable EMA partition. This routine must be called from a program that uses the shareable EMA partition to be locked. The calling sequence is:

```
          EXT LKEMA
           :
           :
          JSB LKEMA
          DEF RTN
      RTN ...
```

This call results in a NOP if the program does not use an EMA, it does not have a shareable EMA, or the shareable EMA is already locked.

Normally, a shareable EMA partition is released once the number of programs actively using it drops to 0. If this partition is locked, it is not released for use by other programs until unlocked either through the ULEMA routine in the system library, or by using a system command UL.

FORTRAN 77: Callable

Pascal: Callable

ULEMA

The ULEMA subroutine unlocks a locked shareable EMA partition. This routine must be called from a program that is using the shareable EMA partition to be unlocked. The calling sequence is:

```
           EXT ULEMA
            :
            :
           JSB ULEMA
           DEF RTN
       RTN ....
```

This call results in a NOP if the program does not use EMA, does not have a shareable EMA, or the shareable EMA partition is not unlocked.

FORTRAN 77: Callable

Pascal: Callable

TATMP

The TATMP subroutine maps the Track Assignment Table (TAT) into the driver partition area in the user's map. The Track Assignment Table (TAT) resides in physical memory and is mapped in the driver area whenever it has to be used. This map will be preserved when the program is swapped. However, when an I/O request is made or a call to one of the EMA routines, (.EMAP, .EMIO, MMAP) is made, the mapping of the driver partition will be destroyed. A call to TATMP must be made if the Track Assignment Table has to be used after an I/O request or a call to the EMA routines is made.

The calling sequence is:

```
          EXT TATMP
           :
           :
          JSB TATMP
          DEF RTN
      RTN ....
```

Pascal: Callable

SEGRT

SEGRT allows a segment to return to the instruction following the SEGLD call in the main program.

CALL SEGRT (Z)
Z is a dummy parameter, supplied to the FORTRAN compiler, indicating a subroutine call. This parameter is not necessary in Pascal, Assembler, or Macro versions of the call.

SEGRT allows any segment that was called from the main by a SEGLD request to return to the instruction following the SEGLD request in the main program.

There are restrictions on the use of SEGRT:

1. SEGRT can only be used if the segment was loaded by a SEGLD request.
2. The segment must have been loaded from the main, not another segment.
3. SEGRT can only return to the main, not another segment.

Chapter 6
Run Time Error Messages

During execution of programs referencing Relocatable Library Subroutines, error messages may be generated. Error messages are listed together with the subroutine involved.

Mathematical Subroutines

Error messages are printed in the form:

program name nn xx

where:

program name	is the name of the user program where the error was encountered.
nn	is a number in the range 02 through 14 which identifies the subroutine involved in the error condition.
xx	is the error type, as follows: OF = Integer or Floating Point Overflow OR = Out of Range UN = Floating Point Underflow

These error messages can occur when system intrinsics are called or during an exponentiation operation. Suppose x and y are real values and I and J are integers. Then, the following relocatable subroutines are called for these computations:

x**y	.RTOR	(real to real)
x**I	.RTOI	(real to integer)
I**J	.ITOI	(integer to integer)

Run Time Error Messages

The following is a summary of possible error messages:

Error Message	Issuing Subroutine	Where Used	Error Condition
02-UN	ALOG	ALOG	$X<0$
		ALOGT	$X\leq 0$
		CLOG	$X=0$
		DLOG	$X<0$
		DLOGT	$X\leq 0$
		.LOG	$X\leq 0$
		.LOG0	$X\leq 0$
		.LOGT	$X\leq 0$
03-UN	SQRT	SQRT	$X<0$
	DSQRT	DSQRT	$X<0$
	.SQRT	.SQRT	

Appendix A
Library Subroutines Callable By Languages

This appendix identifies the library subroutines that can be called by FORTRAN 4X, FORTRAN 77, and Pascal. Where the subroutine is called by a FORTRAN function, the subroutine is listed with (f) following the subroutine name.

FORTRAN 4X

Mathematical Subroutines

ABS (f)
AIMAG (f)
AINT (f)
ALOG (f)
ALOGT (f)
AMOD (f)
ATAN (f)
ATAN2 (f)
CABS (f)
CEXP (f)
CLOG (f)
CMPLX (f)
CONJG (f)
COS (f)
CSNCS (f)
CSQRT (f)
DABS (f)
DATAN (f)
DTAN2 (f)
DBLE (f)
DCOS (f)
DDINT (f)
DEXP (f)
DIM (f)
DLOG (f)
DLOGT (f)
DMOD (f)
DSIGN (f)
DSIN (f)
DSQRT (f)
DTAN (f)
DTANH (f)
ENTIX
EXP (f)
FLOAT (f)
IABS (f)
IAND (f)
IDIM (f)
IDINT (f)
IFIX (f)
INT (f)
IOR (f)
ISIGN (f)
IXOR (f)
MOD (f)
MXMND (f)
MXMNI (f)
MXMNR (f)
REAL (f)
SIGN (f)
SIN (f)
SNGM (f)
SQRT (f)
TAN (f)
TANH (f)
DPOLY (f)
.ABS (f)
.ATAN (f)
.ATN2 (f)
.BLE (f)
.EXP (f)
.LOG (f)
.LOGO (f)
.MOD (f)
.MXMN (f)
.SNCS (f)
.SQRT (f)
.TAN (f)
.TANH (f)
.TSCS (f)
.YINT (f)

Double Integer Subroutines

FIXDR (f)
FLTDR (f)

Utility Subroutines

ABREG
CLRIO
IGET (f)
ISSR (f)
ISSW (f)
MAGTP
NAMR
OVF
PNAME
PTAPE
RMPAR
/TINT (f)

FORTRAN 77

Mathematical Subroutines

ABS (f)
AIMAG (f)
AINT (f)
ALOG (f)
ALOG (f)
AMOD (f)
ATAN (f)
ATAN2 (f)
CABS (f)
CEXP (f)
CLOG (f)
CMPLX (f)
CONJG (f)
COS (f)
CSNCS (f)
CSQRT (f)
DABS (f)
DATAN (f)
DTAN2 (f)
DBLE (f)
DCOS (f)
DDINT (f)
DEXP (f)
DIM (f)
DLOG (f)
DLOGT (f)
DMOD (f)
DSIGN (f)
DSIN (f)
DSQRT (f)
DTAN (f)
DTANH (f)
ENTIX
EXP (f)
FLOAT (f)
IABS (f)
IAND (f)
IDIM (f)
IDINT (f)
IFIX (f)
INT (f)
IOR (f)
ISIGN (f)
IXOR (f)
MOD (f)
MXMND (f)
MXMNI (f)
MXMNR (f)
REAL (f)
SIGN (f)
SNGL (f)
SNGM (f)
SQRT (f)
TAN (f)
TANH (f)
DPOLY (f)
XADSB
XPOLY
.ABS (f)
.ATAN (f)
.ATN2 (f)
.BLE (f)
.CADD
.CDBL
.CDIV
.CFER
.CINT
.CMPY
.CSUB
.CTBL
.CTOI
.DCPX
.DFER
.DINT
.DLD
.DST
.DTOI
.DTOR
.EXP (f)
.ITOI
.LOG (f)
.LOGO (f)
.MAC
.MOD (f)
.MXMN (f)
.NGL
.RTOD
.RTOI
.RTOR
.RTOT
.SIGN
.SNCS (f)
.SQRT (f)
.TAN (f)
.TANH (f)
.TCPX
.TENT
.TINT
.TMTH
.TSCS (f)
.TTOI
.TTOR
.TTOT
.XCOM
.XDIV
.XMPY
.YINT (f)
..CCM
..DCM
..DLC
..FCM
..TCM

Double Integer Subroutines

FIXDR (f)
FLTDR (f)
.DDS
$LOG
$LOGT
$SETP
$SQRT
%ABS
$TAN
%AN
%AND
%ANH
%BS
%FIX
%IGN
%IN
%INT
%LOAT
%LOG
%LOGT
%NT
%OR

%OS	/EXP	PRTN
%OT	/LOG	IFBRK
%QRT	/LOGO	IDGET
%SIGN	/SIN	TMVAL
%SSW	/SQRT	EQLU
%TAN	/TAN	TRMLU
%XP	/TINT (f)	IFTTY
.ENTC		LOGLU
.ENTR	Library Subroutines	LUTRU
.FMUI		LUSES
.FMUO		GTERR
.FMUP	REIO	PTERR
.FMUR	BINRY	SESSN
.GOTO	RNRQ	ICAPS
.MAP	LURQ	SYCON
.OPSY	INPRS	SEGLD
.PCAD	$CVT3	GTSCB
.RCNG	$CVTI	LIMEM
.TAPE	MESSS	CLRQ
/ATLG	FTIME	LKEMA
/COS	GETST	TATMP

Pascal

Mathematical Subroutines

AMOD	DTAN	.ATN2
ATAN2	DTANH	.BLE
CABS	ENTIX	.CADD
CEXP	IAND	.CDBL
CLOG	IDIM	.CDIV
CMPLX	IDINT	.CFER
CONJG	IOR	.CINT
CSNCS	ISIGN	.CMPY
CSQRT	IXOR	.CSUB
DABS	MOD	.CTBL
DATAN	MXMND	.CTOI
DTAN2	MXMNI	.DCPX
DBLE	MXMNR	.DFER
DCOS	REAL	.DINT
DDINT	SIGN	.DLD
DEXP	SNGL	.DST
DIM	SNGM	.DTOD
DLOG	TAN	.DTOI
DLOGT	DPOLY	.DTOR
DMOD	XADSB	.EXP
DSIGN	XPOLY	.ITOI
DSIN	.ABS	.LOG
DSQRT	.ATAN	.LOGO

.MAC
.MOD
.MXMN
.NGL
.RTOD
.RTOI
.RTOR
.RTOT
.SIGN
.SQRT
.TAN
.TANH
.TCPX
.TENT
.TINT
.TSCS
.TTOI
.TTOR
.TTOT
.XCOM
.XDIV
.XMPY
.YINT
..CCM
..DCM
..DCL
..TCM

Double Integer Subroutines

.DDS

Utility Subroutines

ABREG
CLRIO
ERR0
IGET
ISSR
MAGTP
NAMR
OVF
PNAME
PTAPE
RMPAR
SREAD
#COS
#EXP
#LOG
#SIN
$EXP
$LOG
$LOGT
$SETP
$SQRT
%ABS
$TAN
%AN
%AND
%ANH
%BS
%FIX
%IGN
%IN
%INT
%LOAT
%LOG
%LOGT
%NT
%OR
%OS
%OT
%QRT
%SIGN
%SSW
%TAN
%XP
.ENTC
.ENTR
.FMUI
.FMUR
.GOTO
.MAP
.OPSY
.PCAD
.RCNG
.TAPE
/ATLG
/COS
/EXP
/LOG
/LOGO
/SIN
/SQRT
/TAN
/TINT

Library Subroutines

REIO
BINRY
RNRQ
LURQ
INPRS
MESSS
.DRCT
FTIME
GETST
PRTN
IFBRK
IDGET
TMVAL
EQLU
TRMLU
IFTTY
LOGLU
LUTRU
LUSES
GTERR
PTERR
ICAPS
SYCON
SEGLD
GTSCB
LIMEN
CLRQ
LKEMA
ULEMA
TATMP

Index

READER COMMENT SHEET

Relocatable Library
Reference Manual

92084-90013 **December 1981**

Update No. ________
(If Applicable)

We welcome your evaluation of this manual. Your comments and suggestions help us improve our publications. Please use additional pages if necessary.

FROM:

Name ______________________________

Company ______________________________

Address ______________________________

____

Phone No. ____________________ **Ext.** __________

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 141 CUPERTINO, CA.

— POSTAGE WILL BE PAID BY —

Hewlett-Packard Company
Data Systems Division
11000 Wolfe Road
Cupertino, California 95014
ATTN: Technical Marketing Dept.

FOLD

FOLD