AD-A068 930 WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER F/G 9/2
THE INTERVAL ARITHMETIC PACKAGE - MULTIPLE PRECISION VERSION.(U)
JAN 79 J M YOHE DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-1908 NL

| OF |
AD
A068930

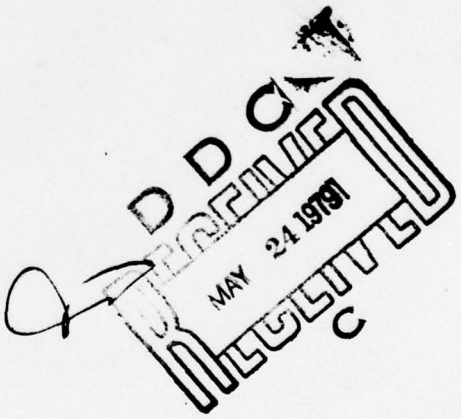
END
DATE
FILMED
6-79
DDC

MRC Technical Summary Report #1908

THE INTERVAL ARITHMETIC PACKAGE - MULTIPLE PRECISION VERSION

J. M. Yohe

Mathematics Research Center
University of Wisconsin—Madison
610 Walnut Street
Madison, Wisconsin 53706



January 1979

Received December 8, 1978

Sponsored by

U. S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

THE INTERVAL ARITHMETIC PACKAGE - MULTIPLE PRECISION VERSION

J. M. Yohe (1)

Technical Summary Report #1908
Jan[redacted]979

ABSTRACT

This report describes the multiple precision version of the interval arithmetic package documented in MRC Technical Summary Report #1755. The multiple precision version, based on the FORTRAN multiple precision arithmetic package of Brent, is extremely portable. It is assumed that the reader has access to TSR #1755, which provides the basic documentation for the interval arithmetic package; the current report addresses only those aspects of the multiple precision version which are not covered in the basic documentation.

AMS(MOS) Subject Classification: 68A10

Key words: Interval Analysis
Interval Arithmetic
Multiple Precision
Extended Precision
Portable Software
Software Package
Precompiler Interface
Augment Interface

Work Unit Number 8 (Computer Science)

16p.

MRC-TSR-1908

(1) Academic Computing Services
The University of Wisconsin - Eau Claire
Eau Claire, Wisconsin 54701

Sponsored by the U. S. Army under Contract No. DAAG29-75-C-0024

221 200

SIGNIFICANCE AND EXPLANATION

Computer calculations are inexact due to rounding error. Interval arithmetic is a method for controlling rounding errors in such a way that an answer, say x, is expressed as a pair of numbers (a,b) between which x must lie, $a \leq x \leq b$.

In machine interval arithmetic it often happens that interval width $b - a$ may grow rapidly due to round-off error. Therefore, in some computations, single precision interval arithmetic may not be adequate to produce satisfactory results.

This paper describes a multiple precision version of the interval arithmetic package documented in MRC Technical Summary Report #1755. The multiple precision version is based on the FORTRAN multiple precision arithmetic package of Brent, which is described in ACM Transactions on Mathematical Software, volume 4 (1978), pp. 57-70. Because of the portability of Brent's package, the multiple precision version of the interval arithmetic package is also extremely portable.

The "standard" precision of the multiple precision version of the interval arithmetic package is about 28 decimal digits. However, the user may elect to use a lower precision by changing certain parameters and constants, or may alter the package to obtain increased precision. This report explains the basic structure of the multiple precision interval package, and provides instructions for its modification.

The responsibility for the wording and views expressed in this descriptive summary lies with MRC, and not with the author of this report.

THE INTERVAL ARITHMETIC PACKAGE - MULTIPLE PRECISION VERSION

J. M. YOHE (1)

1. Introduction:

Because interval arithmetic provides rigorous bounds on the results of computations, it is necessary when implementing the interval algorithms on a computer to round the left endpoint to the nearest machine-representable number less than or equal to the true result of the calculation, and the right endpoint to the nearest machine number greater than or equal to the true result. While this does not seem, on the surface, to be particularly significant, in long computations the widths of intervals may grow quite large due solely to round-off error, even in cases where such inherent problems as dependency are not present. (The classic dependency problem arises from the inherent inability of interval mathematics to recognize cases where the same interval appears in more than one place in an expression; this is discussed elsewhere (see, e.g., [6]), and is not germane to the present discussion.) Thus, in some applications, single precision interval arithmetic may not be adequate.

The interval arithmetic package described in [7], although initially written as a single precision package for the UNIVAC 1100 series computers, was designed for transportability and ease of modification. When the multiple precision arithmetic package of Brent ([1]) appeared, it seemed logical to provide a version of the interval arithmetic package based on that multiple precision arithmetic package. Accordingly, the multiple precision arithmetic package was interfaced with the AUGMENT precompiler ([3], [4], [5]), upon which the original version of the interval package depends; this interface is described in [2]. The interval arithmetic package was then adapted to use the Brent package rather than machine single precision format. This paper describes that adaptation.

In Section 2, we describe the procedure used in adapting the original package to produce the multiple precision version. Section 3 describes the procedure for bringing the multiple precision version up on a host computer other than the UNIVAC 1100 series. Section 4 contains instructions for changing the precision of the package, both statically and dynamically. Section 5 provides instructions for using the multiple precision version of the interval package. The Appendices give the AUGMENT description deck for the multiple precision version of the interval package (Appendix 1) and general guidance for setting up the job control language for a production run of a program which uses the package (Appendix 2).

This paper is intended to be used in conjunction with the basic documentation for the interval arithmetic package [7]. Questions concerning the operations and functions available, the mathematical basis for interval arithme-

(1) Academic Computing Services
The University of Wisconsin - Eau Claire
Eau Claire, Wisconsin 54701

Sponsored by the U. S. Army under Contract No. DAAG29-75-C-0024.

tic, the archtecture of the interval package, etc. should be answered in that document. Here, we address only those aspects which are unique to the multiple precision version.

2. Adapting the interval package:

Five separate steps were required to adapt the original interval arithmetic package for use with the Brent multiple precision arithmetic package. These were: (a) modifying the description deck; (b) writing the BPA primitive routines; (c) modifying the BPA portion of the package; (d) modifying the interval portion of the package; and (e) modifying the test program. These steps are discussed in greater detail below.

Before going into detail, however, a few words about the general philosophy of the adaptation are in order. Since the Brent package provides not only arithmetic, but all appropriate standard mathematical functions as well, we wished to use the Brent package in evaluating the standard functions for interval arithmetic. The basic interval package architecture, however, requires that routines be available to evaluate the standard functions in higher precision than that of the interval endpoints. Moreover, the Brent package did not provide directed roundings in the basic arithmetic operations, and these are required by the interval package. Since the basic arithmetic operations had to be rewritten anyhow, it seemed appropriate to use the Brent package in the role of EXTENDED arithmetic (see [7]), making the interval endpoint arithmetic parallel the arithmetic of the Brent package, but with fewer significant digits. Because of necessity for precise input/output conversions, it was also regarded as desirable to restrict the exponent range of the interval endpoints (BPA numbers). Accordingly, we take the BPA numbers to have three fewer significant digits than the EXTENDED numbers, and restrict the exponent range to numbers in the range of approximately 10**-350 to 10**350.

These remarks having been made, we are now in a position to discuss the details of the modification.

The Description Deck:

The modifications to the description deck were straightforward. EXTENDED was declared as type MULTIPLE rather than DOUBLE PRECISION, and, since this is a nonstandard data type, a copy routine had to be named in the declarations. BPA was then declared as an integer array of length 9 (three less than the declaration for MULTIPLE), and the lines which named unary minus and copy operators were enabled by taking them out of the comment mode. Field functions allowing access to the sign, exponent, and digits of BPA numbers were added, and conversion functions which had named type DOUBLE PRECISION were changed to name type MULTIPLE. The modifications consisted of 25 lines, 10 of which were editing instructions.

The BPA Primitives:

These consisted of the four arithmetic operators plus conversion from EXTENDED to BPA (with directed rounding). The ADD, SUBTRACT, and MULTIPLY routines were patterned after Brent's routines, and there are consequently several supporting routines which are not evident (or even accessible) to the user. The rounding routine in Brent's package was altered to provide directed roundings, but otherwise is quite similar to Brent's. The DIVIDE routine had to be written from scratch, since Brent's package performed division by taking the reciprocal of the divisor and multiplying; this is faster, but does not yield the precision desired for BPA arithmetic.

The only other primitive required for this implementation was a revised version of Brent's MPINIT -- a routine which initializes the multiple precision package. The revised version also initializes the BPA package, including setting of the precision, the exponent range, and calculation of the constants needed for the BPA and INTERVAL portions of the interval package.

Throughout the BPA primitive routines, we have used a COMMON block called BPAMCM, which corresponds exactly with the blank COMMON used by Brent's package (except, of course, that BPAMCM contains the corresponding parameters for the BPA numbers). As we shall see, BPAMCM must be altered whenever the precision of the package is to be increased.

The BPA primitives required about 800 lines of FORTRAN code; however, these primitives should not need to be modified except in the case that the precision is to be increased, and in that case only the COMMON declarations would need to be altered.

Modifying the BPA Portion of the Package:

The modifications to the BPA portion of the package consisted of several parts: The conversion arrays had to be expanded to accommodate the higher precison and larger exponent range; the constants were deleted (since they are now calculated by MPINIT), and various routines which depend on EXTENDED or BPA formats were modified -- notably, the routines which pack and unpack BPA numbers and convert between BPA and other data types. In all, about 100 lines, approximately one quarter of which were editing instructions, were required to perform the modifications.

Modifying the Interval Portion of the Package:

Aside from deleting constant definitions (now set by MPINIT) and changing certain routines which had been DOUBLE PRECISION FUNCTIONS to SUBROUTINES, the majority of the modifications here were concentrated in the INTRAP routine, where output formats and lists had to be altered to conform to the multiple precision format.

Two subroutines were added to the interval portion of the package. The first of these is INTERS, which has as its only argument an integer indicating the number of errors that are allowed before execution is terminated (unless

the INTRAP option calls for unconditional termiantion). This routine is called by the user's program, if it is used at all. The second of these is INTWLK, whose task it is to print a back-trace of subroutine calls in the event of an error. This routine is machine specific, but must have two arguments; the first being an identifier for the calling routine, and the second an integer which is 0 if execution is not to be terminated and nonzero if the error counter is to be interrogated. If the error counter is interrogated, it is decremented by 1; execution is terminated when the error counter is less than or equal to zero.

In all, approximately 100 lines were required for the modification; about one third of these were editing instructions.

Test Program Modification:

About 140 lines, approximately 50 of which were editing instructions, were required for the modification of the test program; nearly all of these were concerned with I/O formats. Some of these lines will need to be revised if the static precision is changed.

3. Bringing the Multiple Precision Interval Package up:

The first step in implementing the multiple precision interval package on any host computer is, of course, to obtain a copy of Brent's multiple precision arithmetic package with the AUGMENT interface (see [2] and [1]). Since this package was not developed at the Mathematics Research Center, it is not appropriate for MRC to distribute it; this package may be obtained from

Dr. Richard P. Brent
Computing Research Group
The Australian National University
Canberra, Australia

Next, one must have a copy of the AUGMENT precompiler. Unless an installation already has AUGMENT, the request for the multiple precision interval arithmetic package should be accompanied by a request for AUGMENT. The documentation supplied with AUGMENT will include the necessary instructions for bringing AUGMENT up on the new host system. Both AUGMENT and the present package may be obtained from

Mathematics Research Center
The University of Wisconsin - Madison
610 Walnut St.
Madison, Wisconsin 53706

With AUGMENT and a copy of Brent's multiple precision arithmetic package available, the next step is to bring up the Brent package. The documentation for that package includes appropriate guidance.

The multiple precision interval arithmetic package, although extremely portable, does require that the user provide three primitive routines: the Hollerith unpacking routine (a similar routine must be provided for the Brent package, and that routine may be used for the interval package if desired, although it must be modified to return a buffer length of 133 if the unpacked string would exceed 132 characters); the error handling routine INTERS (for ERror Set), which simply records the number of nonfatal errors allowed before execution is to be terminated; and the routine INTWLK which, in the UNIVAC 1100 version, prints a trace of the subroutine calls followed by a line of asterisks. The latter routine may be implemented as a routine which merely prints a line of asterisks, if the call trace is not easily implemented or is not wanted.

Once the primitive routines have been provided, it is necessary only to process the multiple precision interval package with the AUGMENT precompiler and then compile it using the standard FORTRAN compiler, placing the compiled modules in a library file. One caveat: we found it necessary to increase the working storage for AUGMENT in order to provide enough work space to handle the descriptions for the multiple precision package and the multiple precision interval package; 5000 words was more than adequate.

The test program should also be AUGMENTed and compiled, then executed to check that the package is working correctly.

If the precision of Brent's multiple precision arithmetic package is to be increased over that defined in its standard AUGMENT interface, certain modifications will also be necessary in the multiple precision interval package. See Section 4 for details.

4. Changing the precision of the package:

Static precision:

By static precision, we mean the maximum allowable precision for a multiple precision interval number. This precision is governed by the number of words of storage allocated to each BPA variable; this is 9 in the standard interface. Lower precision may be used dynamically (see below), although if lower precision is to be used as a standard mode, it would pay to revise the package's standard precision. Higher precision requires altering the dimensions of certain arrays. Increasing the precison may be necessary, for example, to obtain the "standard precision" of about 28 decimal digits on short word-length machines, or to increase precision beyond the "standard" precision.

The only change in the description deck is in Line DSB00020, where the dimension of the integer array for a BPA number must be changed from 9 to n, where n is two more than the number of internal base digits desired. See the documentation for Brent's package for details. The only thing to keep in mind here is that n should be at least three less than the corresponding value for a multiple precision number.

The major change in the BPA primitives is to alter the length of the R array in the COMMON block BPAMCM. An array of length 4n + 10 is adequate. This array is referenced in the following lines of the BPA primitives:

BPP00240
BPP00810
BPP01350
BPP01980
BPP02980
BPP03990
BPP04450
BPP05220
BPP06890

If an exponent range larger than the standard one is desired, line BPP07180 must be changed to the appropriate exponent (base 10). In this case, do not neglect to make the appropriate changes in the conversion arrays in the BPA package (see below).

The COMMON block BPAMCM also appears in certain routines in the BPA portion of the package. Changes must be made in the following lines:

BPA01452
BPA01724
BPA05610
BPA10580

In order to allow working space for the input/output conversion of an exponent greater than about 350 in absolute value, the lengths of the conversion arrays ICX and ECX must be increased, and the position of the radix point must be altered. See [7] for details. In order to do this, changes must be made in the following lines:

BPA00430
BPA00480
BPA00490
BPA01870
BPA03080

In the Interval portion of the package, the COMMON block BPAMCM appears on Line INT11696. Also, in Line INT12920, the number 18 must be changed to 2n, and in Line INT14290, the two occurrences of 9I6 must be changed to nI6 (if n is large enough, this entire format statement may need to be rewritten).

In the test program, BPAMCM is referenced in Line TST12400. No other changes are required in the test program, although many of the formats are too short even to display the full significance present in the "standard" precision.

Dynamic precision:

Dynamic, or run-time, precision may be altered at will, within the bounds of the static precision. However, there are several things to consider:

1) The precision of a BPA number must be no greater than three digits less than the precision of a multiple precision number.

2) If precision is to be increased, one must be certain that the digits beyond those already used in each BPA number (and, by extension, each interval endpoint) are zero.

3) If precision is to be decreased, special care must be taken to assure that the new intervals contain the higher precision intervals. This is true of the constants in the package as well as those intervals that may have been computed.

In this event, we suggest writing a short routine which will round the given intervals to intervals of shorter precision. This might be accomplished as follows:

a) convert the left endpoint to multiple precision.

b) decrease the precision of BPA numbers to the new precision.

c) set OPTION in COMMON block BPACOM to the value of RDL, to enable downward directed rounding.

d) use BPACEB to round the left endpoint.

e) set unused digits of the left endpoint to zero.

f) repeat for the right endpoint, using the value of RDU in the step corresponding to c).

Once these precautions have been taken, change the value of T in COMMON block BPAMCM to n-2, i.e., the number of digits in the shortened precision.

5. Using the multiple precision interval package:

The only unusual feature in the use of the multiple precision interval package is that the user must place the statement

```
INITIALIZE MP
```

before any statement involving multiple precision (EXTENDED), BPA, or INTERVAL variables. AUGMENT will translate this statement to a call on the MPINIT routine, which initializes both the multiple precision package and the interval package. (Note that the above statement will cause AUGMENT to declare MP as a variable.) Care must be taken, of course, to insure that the version of the MPINIT routine supplied with the multiple precision interval package is the one that is linked with the program, instead of the version supplied with the multiple precision arithmetic package of Brent.

Otherwise, the use of the multiple precision package is exactly the same as the use of the single precision version; see [7] for details.

6. Conclusion:

We have attempted to give a concise description of the multiple precision version of the interval arithmetic package, showing how it was derived from the single precision version, how it is implemented on another host system, and how it may be used.

This document is intended to be a supplement to [7], and that report should still be considered the primary source of information for this package. Naturally, the disclaimer stated in that document applies to this version as well.

Suggestions, comments, or corrections may be addressed to the author.

REFERENCES

1. Richard P. Brent, A FORTRAN multiple-precision arithmetic package, Assoc. Comput. Mach. Trans. Math. Software 4 (1978), 57-70.

2. Richard P. Brent, Judith A. Hooper, and J. M. Yohe, An AUGMENT interface for Brent's multiple precision arithmetic package, The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report #1868, August, 1978.

3. F. D. Crary, The Augment precompiler I: User information, The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report #1469, December, 1974 (revised April, 1976).

4. F. D. Crary, The AUGMENT precompiler II: Technical Documentation, The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report #1470, October, 1975.

5. F. D. Crary, A versatile precompiler for nonstandard arithmetics, Assoc. Comput. Mach. Trans. Math. Software (to appear).

6. Ramon E. Moore, Interval Analysis, Prentice - Hall, Inc., Englewood Cliffs, NJ, 1966.

7. J. M. Yohe, The INTERVAL arithmetic package, The University of Wisconsin - Madison, Mathematics Research Center, Technical Summary Report #1755, June, 1977 (revised September, 1977).

APPENDIX 1

DESCRIPTION DECK FOR MULTIPLE PRECISION INTERVAL PACKAGE

```
*DESCRIBE EXTENDED                                                    DSE00010
DECLARE MULTIPLE, KIND SAFE SUBROUTINE, PREFIX EXT                    DSE00020
COMMENT DUMMY DESCRIPTION SO CONVERT CAN BE USED                      DSE00030
SERVICE COPY (STR)                                                    DSE00040
*DESCRIBE BPA                                                         DSB00010
DECLARE INTEGER(9), KIND SAFE SUBROUTINE, PREFIX BPA                  DSB00020
OPERATOR +  (, NULL UNARY, PRV, $)                                    DSB00030
OPERATOR -  (NEG, UNARY, PRV, $)                                      DSB00070
OPERATOR +  (ADD, BINARY 1, PRV, $, COMM), * (MUL), - (SUB,,,,NONCOMM), DSB00080
        /  (DIV)                                                      DSB00090
OPERATOR ** (XBI, BINARY 3, PRV, $, INTEGER, $)                       DSB00100
OPERATOR .EQ. (EQ, BINARY 2, PRV, $, LOGICAL), .NE. (NE), .LT. (LT),  DSB00110
        .LE. (LE), .GT. (GT), .GE. (GE)                               DSB00120
FUNCTION ABS (ABS, ($), $), SIN (SIN), COS (COS), TAN (TAN), ASIN (ASN),DSB00130
        ACOS (ACS), ATAN (ATN), LOG10 (LOG), LOG (LN), LN (),         DSB00140
        EXP (EXP), SINH (SNH), COSH (CSH), TANH (TNH), SQRT (SQT),    DSB00150
        CBRT (CBT)                                                    DSB00160
FUNCTION INT (INT, ($), $)                                            DSB00170
FUNCTION MAX (MAX, ($, $), $), MIN (MIN)                              DSB00180
SERVICE COPY(STR)                                                     DSB00230
FIELD SGN (=MPSIGA, =MPSIGB, ($), INTEGER), EXPON (=MPEXPA, =MPEXPB),  DSB00234
      DIGIT (=MPDGA, =MPDGB, ($, INTEGER))                            DSB00236
CONVERSION CTB (CEB, MULTIPLE, $, DOWNWARD),                          DSB00240
           CTB (CRB, REAL, $, UPWARD),                                DSB00250
           CTB (CIB, INTEGER, $, UPWARD),                             DSB00260
           CTE (CBE, $, MULTIPLE, UPWARD),                            DSB00270
           CTR (CBR, $, REAL, DOWNWARD),                              DSB00280
           CTI (CBI, $, INTEGER, DOWNWARD)                            DSB00290
*DESCRIBE INTERVAL                                                    DSX00010
COMMENT  IN THIS DESCRIPTION DECK, 'MULTIPLE' IS USED AS A            DSX00020
 SYNONYM FOR 'EXTENDED'.  IF EXTENDED IS A TYPE OTHER THAN MULTIPLE,  DSX00030
 THE APPROPRIATE CHANGES WILL NEED TO BE MADE IN THIS DECK.           DSX00040
DECLARE BPA(2), KIND SAFE SUBROUTINE, PREFIX INT                      DSX00080
OPERATOR +  (,NULL UNARY, PRV, $)                                     DSX00090
OPERATOR -  (NEG, UNARY, PRV, $)                                      DSX00100
OPERATOR +  (ADD, BINARY1, PRV, $, COMM), * (MUL), - (SUB,,,,NONCOMM), DSX00110
        /  (DIV)                                                      DSX00120
OPERATOR ** (XXI, BINARY 3, PRV, $, INTEGER, $), ** (XXB,,,,BPA),     DSX00130
        ** (XXE,,,,DOUBLE PRECISION), ** (XXX,,,,$)                   DSX00140
OPERATOR .VEQ. (VEQ, BINARY 2, .EQ. , $, LOGICAL, COMM), .VNE. (VNE), DSX00150
        .SEQ. (SEQ), .SNE. (SNE)                                      DSX00160
OPERATOR .VLT. (VLT, BINARY 2, .LT. , $, LOGICAL), .VLE. (VLE),       DSX00170
        .VGT. (VGT), .VGE. (VGE), .SLT. (SLT), .SLE. (SLE),           DSX00180
        .SGT. (SGT), .SGE. (SGE)                                      DSX00190
OPERATOR .UNION. (UNN, BINARY 1, .OR., $), .INTSCT. (SCT,, .AND.)     DSX00200
OPERATOR .SUBSET. (SBS, BINARY 2, .EQ., $, LOGICAL), .SPRSET. (SPS)   DSX00210
OPERATOR .E. (ELE, BINARY 3, .EQ., BPA, $, LOGICAL)                   DSX00220
```

```
FUNCTION ABS (ABS, ($), $), SIN(SIN), COS (COS), TAN (TAN), ASIN (ASN), DSX00230
         ACOS (ACS), ATAN (ATN), LOG10 (LOG), LN (LN), EXP (EXP),       DSX00240
         SINH (SNH), COSH (CSH), TANH (TNH), SQRT (SQT), CBRT(CBT),     DSX00250
         LOG (LN)                                                       DSX00260
FUNCTION ATAN2 (AT2, ($, $), $)                                         DSX00270
FUNCTION OK  (OK, ($), LOGICAL), BAD (BAD)                              DSX00280
FUNCTION COMPOS (CPS, (BPA, BPA), $)                                    DSX00290
FUNCTION INT (INT, ($), $)                                              DSX00300
FUNCTION DIST (DST, ($, $), BPA)                                        DSX00310
FUNCTION MDPT (MDB, ($), BPA), HLGTH (HLB), LENGTH (LGB), LGTH (LGB),   DSX00320
         MAG (MAG), MIG (MIG), PIVL (PVL), PIVU (PVU), SIZE (SIZ)       DSX00330
FUNCTION SGN (SGN, ($), INTEGER)                                        DSX00340
FUNCTION BOUND (BND, (DOUBLE PRECISION, INTEGER), $)                    DSX00350
FIELD SUP (SUP, SPL, ($), BPA), INF (INF, INL)                          DSX00360
SERVICE COPY (STR)                                                      DSX00370
CONVERSION CTX (CIX, INTEGER, $, UPWARD),                               DSX00380
           CTX (CBX, BPA, $, UPWARD),                                   DSX00390
           CTX (CRX, REAL, $, UPWARD),                                  DSX00400
           CTX (CEX, MULTIPLE, $, UPWARD),                              DSX00410
           CTX (ASG, HOLLERITH, $, UPWARD),                             DSX00420
           CTI (CXI, $, INTEGER, DOWNWARD),                             DSX00430
           CTB (CXB, $, BPA, DOWNWARD),                                 DSX00440
           CTR (CXR, $, REAL, DOWNWARD),                                DSX00450
           CTE (CXE, $, MULTIPLE, DOWNWARD)                             DSX00460
```

APPENDIX 2

RUNSTREAM FOR MULTIPLE PRECISION INTERVAL PACKAGE

```
[INVOKE AUGMENT]
<description deck for Brent's package>
<description deck for multiple precision interval package>
*BEGIN
*CONVERT EXTENDED - MULTIPLE
<main program> (does not apply if package is being compiled)
     <nonexecutable statements>
     INITIALIZE MP
     <executable statements>
<subprogram(s)>
*END
[COMPILE PREPROCESSED PROGRAM MODULES]
<any other processing, such as compilation of other modules,
 file preparation, etc.>
[INVOKE LINKAGE EDITOR]
<special instructions to include BLOCK DATA modules
 BPACOM and INTCCOM>
[EXECUTE PROGRAM]
<data> (if any)
[END OF JOB]
```

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 1908	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE INTERVAL ARITHMETIC PACKAGE - MULTIPLE PRECISION VERSION		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) J. M. Yohe		8. CONTRACT OR GRANT NUMBER(s) DAAG29-75-C-0024
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin 610 Walnut Street Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 8 - Computer Science
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office P.O. Box 12211 Research Triangle Park, North Carolina 27709		12. REPORT DATE January 1979 13. NUMBER OF PAGES 12
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Interval Analysis
Interval Arithmetic
Multiple Precision
Extended Precision
Portable Software
Software package
Precompiler Interface
Augment Interface

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report describes the multiple precision version of the interval arithmetic package documented in MRC Technical Summary Report #1755. The multiple precision version, based on the FORTRAN multiple precision arithmetic package of Brent, is extremely portable. It is assumed that the reader has access to TSR #1755, which provides the basic documentation for the interval arithmetic package; the current report addresses only those aspects of the multiple precision version which are not covered in the basic documentation.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)