LEGIBILITY NOTICE

A major purpose of the Technical Information Center is to provide the broadest dissemination possible of information contained in DOE's Research and Development Reports to business, industry, the academic community, and federal, state and local governments.

Although a small portion of this report is not reproducible, it is being made available to expedite the availability of information on the research discussed herein.

UPGRADE OF THE GRESS (TEST VERSION D) PRECOMPILER TO
ALLOW TRIPLY-DIMENSIONED ARRAYS

J. E. Horwedel

F. G. Pin*

Date Published - December 1986

Computing and Telecommunications Division
at
Oak Ridge National Laboratory
Post Office Box X
Oak Ridge, TN 37830

---

*Engineering Physics and Mathematics Division

Martin Marietta Energy Systems
Operating the
Oak Ridge National Laboratory   Oak Ridge Gaseous Diffusion Plant
Oak Ridge Y-12 Plant   Paducah Gaseous Diffusion Plant
Under Contract No. DE-AC05-84OR21400
for the
U. S. DEPARTMENT OF ENERGY

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

MASTER

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGMENTS	v
ABSTRACT	vii
1. INTRODUCTION	1
2. GRESS UPGRADE.	1
2.1 OPERATION CODES	1
2.2 TESTING	4
3. CPU TIME COMPARISON.	6
4. CONCLUSION.	10
REFERENCES	11
APPENDIX. MODIFICATION TO THE GRESS PRECOMPILER AND LIBRARY TO ALLOW TRIPLY-DIMENSIONED ARRAYS	12

ABSTRACT

GRESS is a FORTRAN precompiler and run time library which automatically processes computer models and adds derivative-taking capabilities to the normal calculated results. The GRESS system is under development at ORNL and work is proceeding in upgrading test versions of the precompiler to allow processing by GRESS of most of the ANSI X3.9-1978 FORTRAN syntax. Heretofore, a major limitation has been the inability of GRESS to process arrays with more than two dimensions. Consequently, a significant amount of human effort was required to convert three- and higher-dimensional arrays into two-dimensional arrays for processing with GRESS. This report describes the addition of ten operation codes to the GRESS precompiler and run time library to allow application to FORTRAN programs with three-dimensional arrays. The new operation codes were tested on a sample problem. A comparison of CPU time is made for sample problems run on a VAX 8600 computer with one-, two-, and three-dimensional arrays. It is recommended that GRESS be further modified to accommodate higher-dimensional arrays up to the ANSI X3.9-1978 limit of seven dimensions.

Table 1. Operation codes added to GRESS to allow triply-dimensional arrays

Mnemonic	Code number	Precision	Routine ^a	Description
LTXACP	54	single	LHSIDE	Load address
LTXCP	55	single	VARREF	Load value
FLTXOP	56	single	VARREF	Float and load value
LVTXACP	57	single	LHSIDE	Load variable address
LVTOP	58	single	VARREF	Load value with variable address
FLVTOP	59	single	VARREF	Float and load value with variable address
DLVTOP	60	double	VARREF	Load double precision value with variable address
DLTXOP	61	double	VARREF	Load double precision value
TDXAOP	62	double	LHSIDE	Load double word address
EVTXAOP	63	double	LHSIDE	Load double word with variable address

^aSubroutine or function in the GRESS precompiler where the operation is specified.

The sample program shown in Fig. 1 has comments to indicate which operation code is used for each of the various FORTRAN statements which include three-dimensional arrays. Note that five of the operation codes are for arrays with fixed dimensions, and five are for arrays with variable dimensions. GRESS locates a variable based on its relative address within a common block. The additional overhead for working with multi-dimensional arrays versus one-dimensional arrays is due to the calculation of the index, or pointer, to the variables storage location within the common block.

The algorithm, which is the basis for the index calculation in all of the two- and three-dimensional operation codes, is essentially the same. For two-dimensional arrays, the index is calculated as:

$$\text{INDEX} = I + I_{\max} (J-1) + \text{OFFSET},$$

where I is the row index, $I_{\max}$ is the row dimension, J is the column index, and OFFSET is the number of words from the beginning of the common block to the beginning of the array.

```

C      PURPOSE- TO TEST GRESS OPERATION CODES 54 THROUGH 63 USED TO
C      PROCESS TRIPLY-DIMENSIONED ARRAYS
C
      REAL*8 RA,RB,RX,RY,DR
      DIMENSION A(3,3,3),B(4,4,4),X(3,3,3),Y(4,3,2),INT(2,2,2),R(3)
      DIMENSION RA(3,3,3),RB(4,4,4),RX(3,3,3),RY(4,3,2),DR(3)
      DATA X/27*3.0/, Y/24*4.0/
      DATA RX/27*3.0D0/, RY/24*4.0D0/
      DATA INT/8*5/
      DATA I2,I3,I4/2,3,4/
C
C      DEFINE X,RX,Y, AND RY AS PARAMETERS TO GRESS--SEE GRESS USERS MANUAL
C
      CALL DEFXX(X(3,2,2))
      CALL DEFXX(Y(4,2,1))
      CALL DEFXX(RX(3,2,2))
      CALL DEFXX(RY(4,2,1))
C
C      GRESS OPCODES 54, 55, AND 56 ARE USED IN THE NEXT LINE
C
      A(2,2,2)=X(3,2,2)*X(3,2,2) + INT(2,2,2)
C
C      GRESS OPCODES 61 AND 62 ARE USED IN THE NEXT LINE
C
      RA(2,2,2)=RX(3,2,2)*RX(3,2,2)
C
      CALL VAR(A,B,X,Y,INT,RA,RB,RX,RY,I2,I3,I4)
C
C      REQUEST DERIVATIVES--SEE GRESS USERS MANUAL
C
      CALL GETNXX(A(2,2,2),1,R(1))
      CALL GETNXX(B(4,4,4),1,R(2))
      CALL GETNXX(B(4,4,4),2,R(3))
      CALL GETNXX(RA(2,2,2),3,DR(1))
      CALL GETNXX(RB(4,4,4),3,DR(2))
      CALL GETNXX(RB(4,4,4),4,DR(3))
C
C      OUTPUT RESULTS AND DERIVATIVES
C
      WRITE(6,10)B(4,4,4),RB(4,4,4)
      WRITE(6,20)R,DR
10      FORMAT(1X,' B = ',F15.1,'      RB = ',D15.2)
20      FORMAT(1X,' DA/DX = ',E12.3,' DB/DX = ',E12.3//
1      ' DB/DY = ',E12.3,' DRA/DRX = ',D12.3//
2      ' DRB/DRX = ',D12.3,' DRB/DRY = ',D12.3)
      STOP
      END
      SUBROUTINE VAR(A,B,X,Y,INT,RA,RB,RX,RY,I2,I3,I4)
C
C      PURPOSE-TO TEST THE VARIABLE ADDRESS GRESS OPERATION
C      CODES 57, 58, 59, 60, AND 63.
C
      REAL*8 RA,RB,RX,RY
      DIMENSION A(I2,I3,I4),B(I4,I4,1),X(I3,I3,1)
      DIMENSION Y(I4,I3,1),INT(I2,I2,1)
      DIMENSION RA(I3,I3,1),RB(I4,I4,1),RX(I3,I3,1)
      DIMENSION RY(I4,I3,1)
C
C      GRESS OPCODES 57, 58, AND 59 ARE USED IN THE NEXT LINE
C
      B(4,4,4)=A(2,2,2) + 5.0*Y(4,2,1) + INT(2,2,2)
C
C      GRESS OPCODES 60 AND 63 ARE USED IN THE NEXT LINE
C
      RB(4,4,4)=RA(2,2,2) + 5.0*RY(4,2,1)
      RETURN
      END

```

Fig. 1. Program used to test the modifications to GRESS to allow triply-dimensioned arrays.

For three-dimensional arrays, the index is calculated as:

$$\text{INDEX} = I + I_{\max} [J + J_{\max} (K - 1) - 1] + \text{OFFSET} ,$$

where $J_{\max}$ is the dimension of the first column, and K is the index of the second column.

The modifications were implemented in a test version of GRESS on a VAX 8600. Note that the actual changes are not machine dependent and can easily be implemented in other versions of GRESS. See the Appendix for a list of the modifications to both the GRESS precompiler and library.

2.2 TESTING

To test the added operations, the sample problem in Fig. 1 was executed. In essence, the problem is solving the equations shown below in both single and double precision.

$$\begin{aligned} A &= X^2 + 5 , \\ B &= A + 5Y + 5 , \end{aligned}$$

with

$$\begin{aligned} X &= 3 , \\ Y &= 4 . \end{aligned}$$

The results, shown in Fig. 2, can be easily verified analytically.

B =	39.0	RB =	0.29D+C2
DA/DX	=	0.600E+01	
DB/DX	=	0.600E+01	
DB/DY	=	0.500E+01	
DRA/DRX	=	0.600D+01	
DRB/DRX	=	0.600D+01	
DRB/DRY	=	0.500D+01	

Fig. 2. Output from program shown in Fig. 1.

3. CPU-TIME COMPARISON

Several test programs (shown in Figs. 3 and 4) were used to investigate the impact of multiply-dimensioned arrays on CPU time. These are singly-, doubly-, and triply-dimensioned versions of the same program. As shown, the test programs were run through GRESS without defining any parameters or requesting any derivatives; however, the additional overhead for multiply-dimensioned arrays is incurred in the addressing of the variables. Increases in CPU time caused by parameter definition, etc., should be the same for all three cases.

a. Singly Dimensioned

```

DIMENSION X(250000),Y(250000)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
KOUNT=KOUNT+1
Y(KOUNT)=X(KOUNT)*9.0
10 CONTINUE
TYPE*,Y(1),Y(250000)
STOP
END

```

b. Doubly Dimensioned

```

DIMENSION X(100,2500),Y(250000)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
KOUNT=KOUNT+1
JK=J+50*(K-1)
Y(KOUNT)=X(I,JK)*9.0
10 CONTINUE
TYPE*,Y(1),Y(250000)
STOP
END

```

c. Triply Dimensioned

```

DIMENSION X(100,50,50),Y(250000)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
KOUNT=KOUNT+1
Y(KOUNT)=X(I,J,K)*9.0
10 CONTINUE
TYPE*,Y(1),Y(250000)
STOP
END

```

Fig. 3. Sample programs with one indexed array reference on the right hand side of the equation for CPU time comparison shown in Table 2.

a. Singly Dimensioned

```

DIMENSION X(250000)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
KOUNT=KOUNT+1
X(KOUNT)=X(KOUNT)*X(KOUNT)
10 CONTINUE
TYPE*,X(1),X(250000)
STOP
END

```

b. Doubly Dimensioned

```

DIMENSION X(100,2500)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
JK=J+50*(K-1)
X(I,JK)=X(I,JK)*X(I,JK)
10 CONTINUE
TYPE*,X(1,1),X(100,2500)
STOP
END

```

c. Triply Dimensioned

```

DIMENSION X(100,50,50)
DATA X/250000*3.0/
KOUNT=0
DO 10 I=1,100
DO 10 J=1,50
DO 10 K=1,50
X(I,J,K)=X(I,J,K)*X(I,J,K)
10 CONTINUE
TYPE*,X(1,1,1),X(100,50,50)
STOP
END

```

Fig. 4. Sample programs with three indexed array references for CPU time comparison shown in Table 3.

The programs shown in Fig. 3 have one indexed variable addressed per equation. The comparison of CPU time is shown in Table 2. Table 2 shows a 17% increase in CPU time in going from one dimension to two dimensions, and a 29% increase in going from one dimension to three dimensions.

Table 2. Comparison of CPU time on the VAX 8600 for the sample programs shown in Fig. 3.

Number of dimensions in array X	Run-time (CPU s)
1	24.0
2	28.2
3	31.0

The test programs shown in Fig. 4 represent a "worst" case scenario for triply-dimensioned arrays. This program has three multiply addressed variables per equation. As shown in Table 3, the increases in CPU time are more significant. The increase in CPU time, as shown in Table 3, in going from one to three dimensions is 59%.

Table 3. Comparison of CPU time on the VAX 8600 for the sample programs shown in Fig. 4.

Number of dimensions in array X	Run-time (CPU s)
1	32.7
2	42.6
3	51.9

Taking into consideration that the GRESS processed program can run in general from 30 to 50 times longer (CPU time) than the unprocessed code, any increases in CPU time caused by multiply-dimensioned arrays can be significant. The results suggest that if a majority of the variable references in a large code are triply-dimensioned, it might be possible to reduce the CPU time by 10 to 50% by converting all multiply-dimensioned array references to singly-dimensioned array references. However, for a large code with a small fraction of its variable references in multiply-dimensioned arrays, only a slight reduction in CPU time can be expected by converting the multiply-dimensioned array references to singly-dimensioned references.

4. CONCLUSION

The changes necessary to allow the processing of triply-dimensioned arrays have been implemented in a test version of GRESS. Results from a sample program indicate the changes are working correctly. Since converting multiply-dimensioned array references to logically equivalent, singly-dimensioned array references can require a significant amount of human effort, the most promising result is in the CPU time comparisons. Only a slight decrease in CPU time can be achieved by converting triply-dimensioned arrays to singly-dimensioned arrays in application of GRESS to codes with a limited number of triply-dimensioned arrays. However, if the majority of the equations in a large code have several triply-dimensioned array references the potential savings might warrant conversion to equivalent singly-dimensioned array references.

It is recommended that future modifications be made to GRESS to allow processing of multiply-dimensioned arrays up to the ANSI X3.9-1978 limit of seven dimensions.⁵ However, since there can be a significant penalty for a large number of multiply-dimensioned array references, it is suggested that GRESS print out a warning message when a significant number of multiply-dimensioned array references are encountered.

REFERENCES

1. E. M. Oblow, GRESS, Gradient-Enhanced Software System, Version D. User's Guide, ORNL/TM-9658, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, July 1985.
2. E. M. Oblow, F. G. Pin, and R. Q. Wright, "Sensitivity Analysis Using Computer Calculus: A Nuclear Waste Isolation Application," Nucl. Sci. Eng., accepted for publication.
3. F. G. Pin, B. A. Worley, E. M. Oblow, and R. Q. Wright, "An Automated Sensitivity Analysis Procedure for the Performance Assessment of Nuclear Waste Isolation Systems," Nucl. Chem. Waste Manag., accepted for publication.
4. F. G. Pin, E. M. Oblow, and R. Q. Wright, Automated Sensitivity Analysis Using the GRESS Language, ORNL/TM-6148, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, December 1977.
5. American National Standard Programming Language FORTRAN, ANSI X 3.9-1978, American National Standards Institute, New York.

APPENDIX. MODIFICATIONS TO THE GRESS (VERSION D) PRECOMPILER AND RUN TIME LIBRARY TO ALLOW PROCESSING OF TRIPLY-DIMENSIONED ARRAYS

Following are the modifications which were made to the GRESS (Version D) precompiler and run time library to allow processing of triply-dimensioned arrays:

Original code:

All occurrences of the following DIMENSION statement should be modified.

```
21      DIMENSION SLOC(9998),STYP(9998),SPDM(9998),SCDM(9998)
```

Modification:

```
      DIMENSION SLOC(9998),STYP(9998),SRDM(9998),SCDM1(9998),SCDM2(9998)
```

Original code: SUBROUTINE CHOOSE

```
633      OFF=MAX0(QBUF(LS+1),1)+RDIM*(MAX0(QBUF(LS+2),1)-1)-1
```

Modification:

```
      CDIM1=MAX0(SCDM1(I),1)
      CDIM2=MAX0(SCDM2(I),1)
      OFF=MAX0(QBUF(LS+1),1)+RDIM*(MAX0(QBUF(LS+2),1)-1)-1
```

Original code: SUBROUTINE CHOOSE

```
659      LENGTH=MAX0(SRDM(REF),1)*MAX0(SCDM(REF),1)
```

Modification:

```
      LENGTH=MAX0(SRDM(REF),1)*MAX0(SCDM1(REF),1)*MAX0(SCDM2(REF),1)
```

Original code: SUBROUTINE CHOCSE

685   M=M*MAX0(SCDM(I),1)

Modification:

M=M*MAX0(SCDM1(I),1)*MAX0(SCDM2(I),1)

Original code: SUBROUTINE CHOOSE

742   SCDM(IS)=SCDM(TRV)

Modification:

SCDM1(IS)=SCDM1(TRV)
SCDM2(IS)=SCDM2(TRV)

Original code: SUBROUTINE CHOOSE

767   SCDM(IS)=0

Modification:

SCDM1(IS)=0
SCDM2(IS)=0

Original code: SUBROUTINE COMDEF

857   CDIM=SCDM(I)

Modification:

```

CDIM1=SCDM1(I)
CDIM2=SCDM2(I)

```

Original code: SUBROUTINE COMDEF

```

879      SCDM( IS)=CDIM

```

Modification:

```

SCDM1( IS)=CDIM1
SCDM2( IS)=CDIM2

```

Original code: SUBROUTINE DECLAR

```

1330      CDIM=SCDM(J)
1331      IF (CDIM.EQ.0) GO TO 40
1332      CALL OUTCH(COMMA)
1333      IF (CDIM.GT.0) CALL OUTNUM(CDIM)
1334      IF (CDIM.LT.0) CALL NBGEN(SNAM(-CDIM),MAXNCH)
1335  40    CALL OUTCH(RPAREN)

```

Modification:

```

      CDIM1=SCDM1(J)
      CDIM2=SCDM2(J)
      IF (CDIM1.EQ.0) GO TO 40
      CALL OUTCH(COMMA)
      IF (CDIM1.GT.0) CALL OUTNUM(CDIM1)
      IF (CDIM1.LT.0) CALL NBGEN(SNAM(-CDIM1),MAXNCH)
      IF (CDIM2.EQ.0) GO TO 40
      CALL OUTCH(COMMA)
      IF (CDIM2.GT.0) CALL OUTNUM(CDIM2)
      IF (CDIM2.LT.0) CALL NBGEN(SNAM(-CDIM2),MAXNCH)
40    CALL OUTCH(RPAREN)

```

Original code: SUBROUTINE DECLAR

```

1377      CDIM=QB'UF(EB+2)
1378      IF (CDIM.EQ.0) GO TO 165
1379      CALL OUTCH(COMMA)
1380      CALL OUTNUM(CDIM)
1381  165   CALL OUTCH(RPAREN)

```

Modification:

```

1368      CDIM1=QB'UF(EB+2)
1369      IF (CDIM1.EQ.0) GO TO 165
1370      CALL OUTCH(COMMA)
1371      CALL OUTNUM(CDIM1)
1372  165   CALL OUTCH(RPAREN)

```

Original code: SUBROUTINE DFSCAL

```

1490      SCDM(1S)=0
1491      SEQU(1S)=0

```

Modification:

```

      SCDM1(1S)=0
      SCDM2(1S)=0
      SEQU(1S)=0

```

Original code: SUBROUTINE DIMENS

```

1613      SCDM(1)=CDIM
1614      IF (CHAR.EQ.COMMA) GO TO 10

```

Modification:

```

      SCDM1(1)=CDIM1
      SCDM2(1)=CDIM2
      IF (CHAR.EQ.COMMA) GO TO 10

```

Original code: SUBROUTINE ENTRY5

1958   SCDM(I)=0

Modification:

SCDM1(I)=0
SCDM2(I)=0

Original code: SUBROUTINE ENTRY5

1976   SCDM(IS)=0

Modification:

SCDM1(IS)=0
SCDM2(IS)=0

Original code: SUBROUTINE EQUIV5

2076   SCDM(I)=0

Modification:

SCDM1(I)=0
SCDM2(I)=0

Original code: SUBROUTINE EQUIV5

2098   QBUF(NEQ-1)=CDIM

Modification:

QBUF(NEQ-1)=CDIM1

Original code: SUBROUTINE HEADER

2722 SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE HEADER

2734 SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE HEADER

2741 SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE HEADER

2772   SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE HEADER

2784   SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE HEADER

2794   SCDM(IS) =0

Modification:

SCDM1(IS) =0
SCDM2(IS) =0

Original code: SUBROUTINE INIT

Modification:

C***** OPERATIONS FOR TRIPLY-INDEXED ARRAYS ADDED BY J.E. HORWEDEL
5/14/86

LTXAOP=54
LTXOP=55
FLTXOP=56
LVTXAOP=57
LVTOP=58
FLVTOP=59
DLVTOP=60
DLTXOP=61
TDXAOP=62
DVTXAOP=63

Original code: SUBROUTINE ITEM

3251   CDIM=0

Modification:

CDIM1=0
CDIM2=0

Original code: SUBROUTINE ITEM

3265   CDIM=-LOOKUP(NAME)
3266   GO TO 40
3267   30   CALL INUM(CDIM)
3268   40   CALL GETC

Modification:

CDIM1=-LOOKUP(NAME)
GO TO 32
30   CALL INUM(CDIM1)
32   IF (CHAR.EQ.RPAREN) GO TO 40
     CALL NEXTC
     IF (TYPE.EQ.TYPEN) GO TO 35
     CALL SYMBOL
     CDIM2=-LOOKUP(NAME)
     GO TO 40
35   CALL INUM(CDIM2)
40   CALL GETC

Original code: REAL FUNCTION LHSIDE

```
3393      IF (CHAR.NE.RPAREN) GO TO 40
3394      J=SRDM(LHSIDE)
```

Modification:

```
J=SRDM(LHSIDE)
IF(CHAR.EQ.COMMA) GO TO 5
IF (CHAR.NE.RPAREN) GO TO 40
```

Original code: REAL FUNCTION LHSIDE

```
3401  10  CALL PUT(K)
```

Modification:

```
      GO TO 10
C TRIPLY-INDEXED
5      IF(J.LE.0)J=SLOC(-J)
      CALL SUBEXP
      IF (FAIL) GO TO 40
      IF (CHAR.NE.RPAREN) GO TO 40
      K=SCDM1(LHSIDE)
      L=LTXAOP
      IF (STYPE.EQ.TYPE8) L=TDXAOP
      IF (K.GT.0) GO TO 7
      L=LVTXAOP
      IF (STYPE.EQ.TYPE8) L=DVTXAOP
      K=SLOC(-K)
7      CALL PUT(L)
10     CALL PUT(K)
```

Original code: SUBROUTINE READC

```
4483      CALL OUTNUM(SCDM(I))
```

Modification:

CALL OUTNUM(SCDM1(I))

Original code: SUBROUTINE RTABLE

```
4710      READ(UNIT) (SNAM(I),SLOC(I),STYP(I),SRDM(I),SCDM(I),SEQU(I),
4711      *      I=1,IS)
```

Modification:

```
      READ(UNIT) (SNAM(I),SLOC(I),STYP(I),SRDM(I),SCDM1(I),SCDM2(I),
*      SEQU(I),I=1,IS)
```

Original code: SUBROUTINE SAVED

```
4771  25    TLOCS(N-4)=RDIM-1+SRDM(I)*(MAX0(CDIM,1)-1)
```

Modification:

```
25    TLOCS(N-4)=RDIM-1+SRDM(I)*(MAX0(CDIM1,1)+(MAX0(CDIM2,1)-1)*
1    SCDM1(I)-1)
```

Original code: SUBROUTINE SAVED

```
4775      TLOCS(N)=SRDM(I)*MAX0(SCDM(I),1)
```

Modification:

```
TLOCS(N)=SRDM(I)*MAX0(SCDM1(I),1)*MAX0(SCDM2(I),1)
```


Original code: SUBROUTINE SETUP

```
4845      LEN=MAX0(1,SRDM(J))*MAX0(1,SCDM(J))
4846      IF (STYP(J).EQ.TYPE8) LEN=LEN+LEN
```

Modification:

```
      LEN=MAX0(1,SRDM(J))*MAX0(1,SCDM1(J))*MAX0(1,SCDM2(J))
      IF (STYP(J).EQ.TYPE8) LEN=LEN+LEN
```

Original code: SUBROUTINE SETUP

```
4855      SCDM( IS)=0
```

Modification:

```
      SCDM1( IS)=0
      SCDM2( IS)=0
```

Original code: SUBROUTINE SETUP

```
4871      SCDM( IS)=0
```

Modification:

```
      SCDM1( IS)=0
      SCDM2( IS)=0
```

Original code: SUBROUTINE SETUP

```
4898      SCDM( IS)=0
```

Modification:

```
SCDM1( IS) =0
SCDM2( IS) =0
```

Original code: SUBROUTINE SETUP

```
4912      SCDM( IS) =0
```

Modification:

```
SCDM1( IS) =0
SCDM2( IS) =0
```

Original code: SUBROUTINE SETUP

```
4938      LEN=MAX0(1,SRDM(J))*MAX0(1,SCDM(J))
```

Modification:

```
LEN=MAX0(1,SRDM(J))*MAX0(1,SCDM1(J))*MAX0(1,SCDM2(J))
```

Original code: SUBROUTINE VARREF

```
5404      SCDM(I)=CDIM
```

Modification:

```
SCDM1(I)=CDIM1
SCDM2(I)=CDIM2
```

Original code: SUBROUTINE VARREF

5470 IF (J.GT.0) GO TO 30

Modification:

IF(CHAR.EQ.COMMA) GO TO 38
IF (J.GT.0) GO TO 30

Original code: SUBROUTINE VARREF

5486 40 CALL PUT(K)

Modification:

GO TO 40
38 IF(J.LE.0) J=SLCC(-J)
CALL SUBEXP
IF(FAIL)RETURN
K=SCDM1(I)
IF(K.GT.0) GO TO 385
K=SLOC(-K)
IF(STYPE-TYPER) 381,382,383
381 L=DLVTOP
GO TO 39
382 L=LVTOP
GO TO 39
383 L=FLVTOP
GO TO 39
385 IF(CTYPE-TYPER) 386,387,388
386 L=DLTXOP
GO TO 39
387 L=LTXOP
GO TO 39
388 L=FLTXOP
39 CALL PUT(L)
40 CALL PUT(K)

Original code: SUBROUTINE WTABLE

```
5544      WRITE(UNIT) (SNAM(I),SLOC(I),STYP(I),SRDM(I),SCDM(I),SEQU(I),  
5545      *      I=1,IS)
```

Modification:

```
      WRITE(UNIT) (SNAM(I),SLOC(I),STYP(I),SRDM(I),SCDM1(I),SCDM2(I),  
      *      SEQU(I),I=1,IS)
```

All modifications were made to SUBROUTINE INTPXX. The following line in the original code should be changed to include the additional line numbers 540000 to 63000, as shown.

42 * 49000,50000,51000,52000,53000),J

Modification:

41 * 49000,50000,51000,52000,53000,54000,55000,56000,
42 * 57000,58000,59000,60000,61000,62000,63000),J

The following section of code should be included just before the RETURN statement in SUBROUTINE INTPXX.

```

C
C-----
C          LTXAOP - LOAD ADDRESS, TRIPLY-INDEXED
54000 N=CODE(IC+2)
      IF (N.LT.0) N=LARG(-N)
      TOP=TOP-1
      IS(TOP-1)=N+IS(TOP-1)+CODE(IC+1)*(CODE(IC)*(IS(TOP+1)-1)+
1      IS(TOP)-1)-1
      IC=IC+3
      TOP=TOP-1
      GO TO 20

C
C-----
C          LTXOP - LOAD, TRIPLY-INDEXED
55000 N=CODE(IC+2)
      IF (N.LT.0) N=LARG(-N)
      N=N+IS(TOP-2)+CODE(IC+1)*(CODE(IC)*(IS(TOP)-1)+IS(TOP-1)-1)-1
      RS(TOP-2)=RDAT(N)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

```

```

C
C-----
C          FLTXXOP - FLOAT AND LOAD, TRIPLY-INDEXED
56000 N=CODE(IC+2)
      IF (N.LT.0) N=LARG(-N)
      N=N+IS(TOP-2)+CODE(IC+1)*(CODE(IC)*(IS(TOP)-1)+IS(TOP-1)-1)-1
      RS(TOP-2)=IDAT(N)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

C
C-----
C          LVTXAOP - LOAD VARIABLE ADDRESS, TRIPLY-INDEXED
57000 N=-CODE(IC+2)
      M=-CODE(IC+1)
      M=LARG(M)
      L=-CODE(IC)
      L=LARG(L)
      TOP=TOP-1
      IS(TOP-1)=LARG(N)+IS(TOP-1)+IDAT(M)*(IDAT(L)*(IS(TOP+1)-1)+IS(TOP)
&          -1)-1
      TOP=TOP-1
      IC=IC+3
      GO TO 20

C
C-----
C          LVTOP - LOAD VARIABLE, TRIPLY-INDEXED
58000 N=-CODE(IC+2)
      M=-CODE(IC+1)
      M=LARG(M)
      L=-CODE(IC)
      L=LARG(L)
      N=LARG(N)+IS(TOP-2)+IDAT(M)*(IDAT(L)*(IS(TOP)-1)+IS(TOP-1)-1)-1
      RS(TOP-2)=RDAT(N)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

C
C-----
C          FLVTOP - FLOAT AND LOAD VARIABLE, TRIPLY-INDEXED
59000 N=-CODE(IC+2)
      M=-CODE(IC+1)
      M=LARG(M)
      L=-CODE(IC)
      L=LARG(L)
      N=LARG(N)+IS(TOP-2)+IDAT(M)*(IDAT(L)*(IS(TOP)-1)+IS(TOP-1)-1)-1
      RS(TOP-2)=IDAT(N)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

```

```

C
C-----
C                               DLVTOP - LOAD DOUBLE VARIABLE, TRIPLY-INDEXED
60000 N=-CODE(IC+2)
      M=-CODE(IC+1)
      M=LARG(M)
      L=-CODE(IC)
      L=LARG(L)
      N=LARG(N)+2*(IS(TOP-2)-IDAT(M)*(IDAT(L)*(IS(TOP)-1)+
&      IS(TOP-1)-1)-1)
      RS(TOP-2)=DDAT(N/2)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

C
C-----
C                               DLTXP - LOAD DOUBLE, TRIPLY-INDEXED
61000 N=CODE(IC+2)
      IF (N.LT.0) N=LARG(-N)
      N=N+2*(IS(TOP-2)+CODE(IC+1)*(CODE(IC)*(IS(TOP)-1)+IS(TOP-1)-1)-1)
      RS(TOP-2)=DDAT(N/2)
      IS(TOP-1)=GRADXX(LOCS(N))
      IC=IC+3
      TOP=TOP-1
      GO TO 20

C
C-----
C                               TDXAOP - LOAD DOUBLE ADDRESS, TRIPLY-INDEXED
62000 N=CODE(IC+2)
      IF (N.LT.0) N=LARG(-N)
      TOP=TOP-1
      IS(TOP-1)=N+2*(IS(TOP-1)+CODE(IC+1)*(CODE(IC)*(IS(TOP-1)-1)+
&      IS(TOP-1)-1)-1)
      TOP=TOP-1
      IC=IC+3
      GO TO 20

C
C-----
C                               DVTXAOP - LOAD DOUBLE ADDRESS VARIABLE, TRIPLY-INDEXED
63000 N=-CODE(IC+2)
      M=-CODE(IC+1)
      M=LARG(M)
      L=-CODE(IC)
      L=LARG(L)
      TOP=TOP-1
      IS(TOP-1)=LARG(N)+2*(IS(TOP-1)+IDAT(M)*(IDAT(L)*(IS(TOP-1)-1)+
&      IS(TOP-1)-1)-1)
      TOP=TOP-1
      IC=IC+3
      GO TO 20
      END

```

INTERNAL DISTRIBUTION

1. R. G. Alsmiller, Jr.	24. E. M. Oblow
2. J. Barhen	25-29. F. G. Pin
3. D. G. Cacuci	30. D. B. Reister
4. A. G. Croff	31. T. H. Row
5. M. G. Culbertson	32. D. Steiner (Consultant)
6. P. W. Dickson, Jr. (Consultant)	33. L. D. Trowbridge
7. J. R. Einstein	34. C. R. Weisbin
8. G. F. Flanagan	35. R. M. Westfall
9. W. E. Ford III	36. B. A. Worley
10. G. H. Golub (Consultant)	37. R. Q. Wright
11. R. M. Haralick (Consultant)	38. A. Zucker
12-16. J. E. Horwedel	39-40. Central Research Library
17. J. K. Ingersoll	41. EP&MD Reports Office
18. R. P. L. inius/G. E. Whitesides	42. Laboratory Records
Central Research Library	43-44. Laboratory Records, ORNL, PC
19. J. L. Lucius	45. ORNL-Y-12 Technical Library
20. P. E. Maerker	Document Reference Section
21. F. C. Maienschein	46. ORNL Patent Office
22. L. J. Mezga	47-48. Radiation Shielding Information
23. F. R. Mynatt	Center

EXTERNAL DISTRIBUTION

49. Donald Alexander, U. S. Department of Energy, Civilian Radioactive Waste Management Program, 1000 Independence Ave., SW, Washington, DC 20585
50. G. Appel, Salt Repository Project Office, 505 King Ave., ONWI-Battelle, Columbus, OH 43201
51. L. C. Brown, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352
52. W. A. Carbiener, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
53. N. F. Carter, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
54. P. M. Clifton, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352
55. C. Cole, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
56. R. Cranwell, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185
57. D. H. Dahlem, U. S. Department of Energy - BWIPO, Richland, WA 99352
58. C. DeFigh-Price, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352

59. P. Doctor, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
60. R. Erikson, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
61. M. Foley, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
62. C. F. Geier, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352
63. S. K. Gupta, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
64. W. Harper, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
65. J. C. Helton, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185
66. R. L. Iman, Sandia National Laboratories, P. O. Box 5800, Albuquerque, NM 87185
67. G. Jansen, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
68. J. F. Kircher, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
69. C. Koplik, The Analytic Sciences Corporation, 1 Jacob Way, Reading, MA 10867
70. P. E. Lamont, U. S. Department of Energy - BWIPO, Richland, WA 99352
71. D. Langmuir, Dept. of Chemistry and Geochemistry, Colorado School of Mines, Golden, CO 80401
72. W. L. Lee, R. F. Weston, Inc., 2301 Research Blvd., Rockville, MD 20850
73. A. Liebetrau, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
74. C. F. McLane, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352
75. E. Meadow, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
76. P. Mohler, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
77. J. C. Neff, U. S. Department of Energy, Richland Operations Office 505 King Ave., Columbus, OH 43201
78. J. Farry, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
79. M. Piepho, Pacific Northwest Laboratory, P. O. Box 999, Richland, WA 99352
80. T. H. Pigford, Department of Nuclear Engineering, University of California, Berkeley, CA 94720
81. G. E. Raines, Battelle Memorial Institute, Office of Nuclear Waste Isolation, 505 King Avenue, Columbus, OH 43201
82. J. Rhoderick, U. S. Department of Energy/OCRWM, 1000 Independence Ave., SW, Washington, DC 20585
83. J. N. Rogers, Div. 8324, Sandia Laboratories, Livermore, CA 94550
84. B. Sagar, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352
85. W. W. Schulz, Rockwell Hanford Company, P. O. Box 800, Richland, WA 99352

86. M. Shupe, Rockwell Hanford Company, P. O. Box 800, Richland,
WA 99352
87. S. Sneider, Pacific Northwest Laboratory, P. O. Box 999, Richland,
WA 99352
88. J. Sonnicksen, Rockwell Hanford Company, P. O. Box 800, Richland,
WA 99352
89. M. K. Thompson, U. S. Department of Energy - BWIPO, Richland, WA 99352
90. G. Underberg, Rockwell Hanford Company, P. O. Box 800, Richland,
WA 99352
91. R. T. Wild, Rockwell Hanford Company, P. O. Box 800, Richland,
WA 99352
92. D. F. Wood, Rockwell Hanford Company, P. O. Box 800, Richland,
WA 99352
- 93-122. Office of Scientific and Technical Information, U. S. Department
of Energy, P. O. Box 62, Oak Ridge, TN 37831
123. ONWI Library, Battelle Memorial Institute, Office of Nuclear
Waste Isolation, 505 King Avenue, Columbus, OH 43201
124. U. S. Department of Energy, Division of Engineering, Mathematics
and Geosciences Branch, Washington, DC 20545
125. U. S. Department of Energy, Oak Ridge Operations, Office of
Assistant Manager for Energy Research and Development, Oak Ridge,
TN 37830