General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)
CDC to CRAY FORTRAN Conversion Manual

Charlotte McGary
Donna Diebert

October, 1983
CDC to CRAY FORTRAN
Conversion Manual
TN-83-7104-802-1

Charlotte McGary
Donna Diebert
Informatics General Corporation
1121 San Antonio Road
Palo Alto, CA 94303

Prepared for
Ames Research Center under
CONTRACT NAS2-11555
October, 1983

NASA
0.0 SCOPE

This document is intended to be an aid in converting source code from FORTRAN IV Extended as used on the CDC 7600 under the SCOPE operating system to FORTRAN 77 as used on the CRAY-1S/CRAY-XMP under COS. Although there are many similarities between CDC and CRAY FORTRAN, there are several differences which can cause significant problems for the user who is not experienced in the use of both. This document discusses many important differences between these two versions of FORTRAN, and gives examples concerning their usage on both computers.

The author would like to acknowledge the following people who contributed to this document: Donna Diebert, Dave Saunders, and Gary Villere, all with Informatics General Corporation.

This work was performed under contract number NAS2-11555, Ames Research Center, by Informatics General Corporation.
Section 1
GENERAL FORTRAN DIFFERENCES BETWEEN
THE CDC 7600 AND THE CRAY

1.0 GENERAL FORTRAN DIFFERENCES BETWEEN THE CDC 7600 AND THE CRAY

While there are many similarities between CDC and CRAY FORTRAN, there are significant differences due to the fact that the CDC 7600 uses FORTRAN IV Extended and the CRAY uses FORTRAN 77. This section will deal with some basic differences between the two.

Diagrams of the general ordering of statements in CDC and CRAY program units are shown in figures 1-1 and 1-2, respectively.

<table>
<thead>
<tr>
<th>CDC</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>60-bit words</td>
<td>64-bit words</td>
</tr>
<tr>
<td>10 characters per word (max)</td>
<td>8 characters per word (max)</td>
</tr>
<tr>
<td>Variable names: 1-7 letters</td>
<td>Variable names: 1-8 letters</td>
</tr>
<tr>
<td>(ANSI FORTRAN standard provides for a maximum of 6 letters)</td>
<td></td>
</tr>
<tr>
<td>Maximum of 3 dimensions for an array</td>
<td>Maximum of 7 dimensions for an array</td>
</tr>
<tr>
<td>Max. array size = 131,071 words</td>
<td>Max. array size = 4,194,304 words</td>
</tr>
</tbody>
</table>

RANGE OF CONSTANTS:

<table>
<thead>
<tr>
<th>CDC</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer: (-2^{59} - 1 &lt; I &lt; 2^{59} - 1)</td>
<td>Integer: (-2^{63} &lt; I &lt; 2^{63})</td>
</tr>
<tr>
<td>18 decimal digits</td>
<td>19 decimal digits</td>
</tr>
<tr>
<td>-293</td>
<td>322</td>
</tr>
<tr>
<td>Real: (10 &lt;</td>
<td>R</td>
</tr>
<tr>
<td>14 decimal digits of precision</td>
<td>14 decimal digits of precision</td>
</tr>
<tr>
<td>-293</td>
<td>322</td>
</tr>
<tr>
<td>Complex: (10 &lt;</td>
<td>C</td>
</tr>
<tr>
<td>real</td>
<td>real imag</td>
</tr>
<tr>
<td>10</td>
<td>2466</td>
</tr>
<tr>
<td>(</td>
<td>C</td>
</tr>
</tbody>
</table>

Hollerith data

- Hollerith data supported but CHARACTER data preferred

No block IF structure is available

- Block IF structure is available

Only one IMPLICIT statement allowed per program unit

- No limit imposed on the number of IMPLICIT statements allowed per program unit
<table>
<thead>
<tr>
<th>OVERLAY statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROGRAM, FUNCTION, SUBROUTINE, or BLOCK DATA statement</td>
</tr>
<tr>
<td>IMPLICIT statement</td>
</tr>
<tr>
<td>type: INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL, COMMON, DIMENSION, EQUIVALENCE, EXTERNAL, and LEVEL statements</td>
</tr>
<tr>
<td>FORMAT statements</td>
</tr>
<tr>
<td>DATA statements</td>
</tr>
<tr>
<td>NAMELIST statements</td>
</tr>
<tr>
<td>Statement Function Definitions</td>
</tr>
<tr>
<td>ENTRY and executable statements</td>
</tr>
<tr>
<td>END statement</td>
</tr>
</tbody>
</table>

Figure 1-1 Order of Statements in a CDC Program Unit
<table>
<thead>
<tr>
<th>ENTRY and FORMAT statements</th>
<th>IMPLICIT statement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>type: INTEGER, REAL, DOUBLE PRECISION, COMPLEX, CHARACTER, LOGICAL, COMMON, DIMENSION, EQUIVALENCE, INTRINSIC, POINTER and SAVE statements</td>
</tr>
<tr>
<td>DATA statements</td>
<td>NAMELIST statements</td>
</tr>
<tr>
<td></td>
<td>Statement Function Definitions</td>
</tr>
<tr>
<td></td>
<td>Executable statements</td>
</tr>
<tr>
<td></td>
<td>END statement</td>
</tr>
</tbody>
</table>

Figure 1-2 Order of Statements in a CRAY Program Unit
Section 2
SPECIFIC FORTRAN DIFFERENCES

2.0 SPECIFIC FORTRAN DIFFERENCES

This section discusses CDC and CRAY differences in the usage of intrinsic functions, program units, arrays, LEVEL and MOVLEV statements, ENTRY statements, and DO loops.

2.1 INTRINSIC FUNCTIONS

Most of the standard intrinsic functions are identical on both the CDC and the CRAY. The following functions are the exception.

<table>
<thead>
<tr>
<th>CDC</th>
<th>No.</th>
<th>Notes/ Limitations</th>
<th>CRAY</th>
<th>No.</th>
<th>Notes/ Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHIFT</td>
<td>2</td>
<td>(0 \leq \text{Arg} \leq 60) (2)</td>
<td>SHIFT</td>
<td>2</td>
<td>None</td>
</tr>
<tr>
<td>Arg &lt;0</td>
<td>2</td>
<td>SHIFTL 2</td>
<td>Zero filled</td>
<td></td>
<td></td>
</tr>
<tr>
<td>((0 \leq</td>
<td>\text{Arg}</td>
<td>\leq 60) for (2) both conditions)</td>
<td></td>
<td>No equivalent right shift is available.</td>
<td></td>
</tr>
<tr>
<td>MASK</td>
<td>1</td>
<td>(0 \leq \text{Arg} \leq 60) (Left justified)</td>
<td>MASK</td>
<td>1</td>
<td>(0 \leq \text{Arg} \leq 63) (Left justified)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>64 (\leq \text{Arg} \leq 128) (Right justified)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOC</td>
<td>1</td>
<td>None</td>
<td>LOC</td>
<td>1</td>
<td>None</td>
</tr>
<tr>
<td>RANF</td>
<td>1</td>
<td>None</td>
<td>RANF</td>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>RANGET</td>
<td>1</td>
<td>None</td>
<td>RANGET</td>
<td>0</td>
<td>None</td>
</tr>
</tbody>
</table>

+ This is a utility subprogram rather than an intrinsic function. It is included here because it corresponds to an intrinsic function.
2.2 PROGRAM UNITS

2.2.1 PROGRAM statement

Although the PROGRAM statement used on the CDC 7600 may be used on the CRAY, the parameters specified after the program name are ignored on the CRAY. The following examples show two CDC uses of the PROGRAM statement and the CRAY equivalent for each.

Example 2-1. Associate INPUT and OUTPUT with logical unit numbers 5 and 6, respectively.

CDC usage: PROGRAM XAMPLI(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

CRAY usage: PROGRAM XAMPLI

or

PROGRAM XAMPLI(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

(Normally the user would assign filenames to logical unit numbers in the CCL. However, the CRAY has been set up at AMES to automatically assign logical unit numbers 5 and 6 to $\text{IN}$ and $\text{OUT}$, respectively.)

Example 2-2. Associate the local file MYFILE with logical unit number 1, which will be used as an input file to the program.

CDC usage: Place the following command in the JCL before the program is executed.

ATTACH,MYFILE,MYFILE,ID=XXX,CY=1.

The FORTRAN PROGRAM statement will be as follows:

PROGRAM XAMPL2(INPUT,OUTPUT,TAPE1=MYFILE)

CRAY usage: Place the following commands in the CCL before the program is executed.

ACCESS,DN=MYFILE,PDN=MYFILE,ID=XXX,ED=1.
ASSIGN,DN=MYFILE,A=PT01.

The FORTRAN PROGRAM statement will follow the same format as in example 2-1.

PROGRAM XAMPL2

or

PROGRAM XAMPL2(INPUT,OUTPUT,TAPE1=MYFILE)
2.2.2 Statement Functions

The STATEMENT FUNCTION is used in basically the same manner and subject to most of the same restrictions on both the CDC 7600 and the CRAY. However, two important differences have been noted and are described below.

STATEMENT FUNCTION NAMES AND EXPRESSIONS

<table>
<thead>
<tr>
<th>CDC usage</th>
<th>CRAY usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATEMENT FUNCTION NAMES AND EXPRESSIONS MUST BE OF THE SAME TYPE.</td>
<td>STATEMENT FUNCTION NAMES AND EXPRESSIONS CAN BE OF DIFFERENT TYPES.</td>
</tr>
<tr>
<td>If a statement function is declared as one type function but is defined with a parameter of a different type, conversion is automatically performed when the function is evaluated. Therefore, the function may be referenced with a different type of variable than that with which it was defined.</td>
<td>Since a statement function name and expression can differ in type, an error occurs if the function is referenced with a different type variable than that with which it was defined. Therefore, the function and variable with which it is referenced must agree in type.</td>
</tr>
</tbody>
</table>

Example 2-3a.
A statement function is declared as INTEGER:

```
INTEGER FCN
```

Statement Function Definition:

```
FCN(R) = R + I * 2
```

Statement Function Reference:

```
NUMBER = FCN(N) + 1
```

Example 2-3b.
A statement function is declared as INTEGER:

```
INTEGER FCN
```

Statement Function Definition:

```
FCN(J) = J + I * 2
```

Statement Function Reference:

```
NUMBER = FCN(N) + 1
```
STATEMENT FUNCTION DEFINITIONS

A statement function definition may reference another statement function even if that function has not previously been defined.

Example 2-4a.

Statement Function Definitions:

MYFCN1(I) = I * MYFCN2(I)
MYFCN2(I) = I + 1

Example 2-4b.

Statement Function Definitions:

MYFCN2(I) = I + 1
MYFCN1(I) = I * MYFCN2(I)

2.3 ARRAYS

2.3.1 Dimension Declarstors

ACTUAL ARRAYS:

On the CDC 7600, the value of a subscript (expressed as a constant) may never be zero or negative. On the CRAY, however, subscripts are allowed to be either zero or negative. Hence, the lower and upper bounds of an array may be expressed in the DIMENSION statement on the CRAY (omission of the lower bound causes the default of 1 to be used). Examples follow.

Example 2-5. Set the variable B equal to the third element of the six-element array A.

CDC usage

DIMENSION A(6)
B = A(3)

CRAY usage

DIMENSION A(6)
B = A(3)

or

DIMENSION A(1:6)
B = A(3)

or

DIMENSION A(-2:3)
B = A(0)
ADJUSTABLE ARRAYS:

Although the CDC 7600 does not allow an array subscript to be zero, the compiler does allow the value of zero to be passed to a subroutine as the dimension declarator of an adjustable array. However, if this array is then used in any calculations within that subroutine, an execution error will occur.

On the CRAY, the dimension declarator(s) must be defined such that the adjustable array will have at least one element in it.

The following example gives a situation on the CDC as discussed above, and the changes necessary to modify that situation to work on the CRAY.

CDC usage

Example 2-6a.

In calling program:

CALL SUB1(A,0,0)

In subroutine SUB1:

SUBROUTINE SUB1(A,N,M)
DIMENSION M(N)

CRAY usage

Example 2-6b.

In calling program:

CALL SUB1(A,1,0)

In subroutine SUB1:

SUBROUTINE SUB1(A,N,M)
DIMENSION M(N)

2.3.2 Type Declaration - Adjustable Arrays

The CDC 7600 compiler allows a formal parameter used as the dimension declarator of an adjustable array to be typed AFTER it is used to dimension the array. This causes a fatal error on the CRAY, however, since the CRAY requires that the type declaration statement for this variable occur BEFORE it is used to dimension the adjustable array. The following example illustrates this difference.

Example 2-7a.

CDC usage

SUBROUTINE MYSUB(A,R,M)
DIMENSION M(R)
INTEGER R

CRAY usage

SUBROUTINE MYSUB(A,R,M)
INTEGER R
DIMENSION M(R)
2.3.3 Array Usage

References to multidimensional arrays may not be shortened on the CRAY as allowed on the CDC 7600.

Example 2-8a.

CDC usage

```plaintext
DIMENSION ARRAY(2,4)
DO 10 I=1,8
  ARRAY(I) = 0.0
10 CONTINUE
```

Example 2-8b.

CRAY usage

```plaintext
DIMENSION ARRAY(2,4)
DO 10 I=1,8
  ARRAY(I,1) = 0.0
10 CONTINUE
```

2.4 LEVEL and MOVLEV Statements

The LEVEL and MOVLEV statements associate variables with Large Core Memory (LCM) on the CDC 7600. There are no equivalent statements on the CRAY, nor are any necessary due to the relatively large amount of memory available on the CRAY. All LEVEL statements should be deleted from CDC programs when converting to the CRAY. The MOVLEV statements may be retained on the CRAY if a dummy MOVLEV routine which just copies N words from one array to another is added to the program.

2.5 ENTRY Statement

Differences in CDC and CRAY usage of entry points are discussed below.

CDC usage

No argument list (assumes the same calling sequence)

CRAY usage

In Subroutine Subprograms:
  Argument list is optional. Arguments need not agree with those specified in FUNCTION, SUBROUTINE, or other ENTRY statements in the same subprogram.
Function entry names must agree in type with the name appearing in the FUNCTION statement of the subprogram in which the ENTRY statement occurs.

In Function Subprograms:
Argument list is required, even if it is null, e.g.
ENTRY NAME()

Function entry names may differ in type from the name appearing in the FUNCTION statement of the subprogram in which the ENTRY statement occurs.

2.6 DO LOOPS

A DO LOOP on both the CDC 7600 and the CRAY takes on the following form:

\[
\text{DO an } i=m_1, m_2, m_3 \\
\text{DO on } i=m_1, m_2, m_3
\]

where
- \( m_1 \) = terminal statement number
- \( m_2 \) = initial value
- \( m_3 \) = terminal value
- \( m_4 \) = optional increment value (default = 1)

Differences between the CDC and CRAY usage of these parameters are discussed below.

<table>
<thead>
<tr>
<th>CDC usage</th>
<th>CRAY usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i ) must be the name of an integer variable with a positive, non-zero value.</td>
<td>( i ) may be the name of an integer, real, or double precision variable.</td>
</tr>
<tr>
<td>( m_1, m_2, m_3 ) must each be the name of an integer variable with a positive, non-zero value such that ( m_1 + m_2 + m_3 &lt;= 2 - 1 )</td>
<td>( m_1, m_2, m_3 ) may be the names of integer, real, or double precision variables.</td>
</tr>
<tr>
<td>If ( m_1, m_2, \text{ and } m_3 ) are integers, ( m_1 + m_2 + m_3 &lt;= 2 - 1 )</td>
<td>If ( m_1, m_2, \text{ and } m_3 ) are integers, ( m_1 + m_2 + m_3 &lt;= 2 - 1 )</td>
</tr>
</tbody>
</table>

If \( m_1, m_2, \text{ and } m_3 \) are integers, then \( m_1, m_2, m_3 \) and \((m_1 - m_2 + m_3) \) must be \( <= 2 - 1 \)
If necessary, \( a_1, a_2, \text{ and/or } a_3 \) are converted to the same type variable as \( i \).

\( a_1, a_2, \text{ and } a_3 \) may be positive or negative.

\( a_3 \) may not be zero.

\begin{align*}
\text{If } a_1 > a_2 \text{ the loop is executed at least once.} \\
\text{If } a_1 > a_2 \text{ and } a_1 > 0 \text{ or } a_2 < a_3 \text{ and } a_3 < 0, \text{ then}
\end{align*}

the loop is not executed unless the \text{ON=J} parameter was specified on the CFT control statement.

This is a significant difference between the CDC and the CRAY treatment of DO LOOPS as it can cause values computed in a CDC program to differ from those output by its CRAY counterpart. In order to obtain the same results from both programs, it is often necessary to force the CRAY to execute each DO LOOP at least once as on the CDC 7600. This can be done by compiling the program with the \text{ON=J} parameter as mentioned above.
Section 3
INPUT/OUTPUT

3.0 INPUT/OUTPUT

3.1 Utilities

Use of the following I/O utilities on the CDC and the CRAY are compared.

<table>
<thead>
<tr>
<th>CDC</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIT(lun)</td>
<td>UNIT(lun)</td>
</tr>
</tbody>
</table>

returns:

-1. transfer successful
+0. EOF encountered
+1. Parity error encountered
+2. Unit error encountered

(CDC returns: +2.0 Unit error encountered)

returns:

-2.0 record partly read
-1.0 transfer successful
0.0 EOF or EOD encountered
+1.0 Parity error encountered
+2.0 Unit error encountered

(use of the END=xxx and ERR=xxx parameters on the READ statement are recommended instead of this utility.)

EOF(lun)

returns:

0. no EOF encountered
non-zero EOF encountered

(Returns: +1.0 EOF encountered)

NW-LENGTH(lun)

returns:

Number of words transferred
from logical unit number
"lun".

(Number of words transferred
(NW) to or from unit "lun".)

If an EOF or EOD is read
from logical unit number
"lun" a zero is returned.

(Applies only to buffered input/output operations.)
3.2 Format Specifications

Following are brief descriptions of differences between CDC and CRAY usage of several format specifications.

<table>
<thead>
<tr>
<th>CDC usage</th>
<th>CRAY usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ow.d w &gt;= d + 6</td>
<td>Ow.d w &gt; d + 4</td>
</tr>
<tr>
<td>Ow Input field may contain a maximum of 20 octal digits.</td>
<td>Ow Input field may contain a maximum of 22 octal digits.</td>
</tr>
<tr>
<td>* and = edit descriptors allowed.</td>
<td>* and = edit descriptors not allowed.</td>
</tr>
<tr>
<td>' edit descriptor is not allowed.</td>
<td>' edit descriptor allowed.</td>
</tr>
<tr>
<td>nH descriptor can be used for input and output.</td>
<td>nH can only be used for output.</td>
</tr>
<tr>
<td>-nX allowed.</td>
<td>-nX not allowed.</td>
</tr>
<tr>
<td>n(//) repeats / n times.</td>
<td>n/ repeats / n times.</td>
</tr>
<tr>
<td>Tn is the only tab specification available.</td>
<td>TLn and TRn are available in addition to Th.</td>
</tr>
<tr>
<td>BN and BZ editing are not available.</td>
<td>BN and BZ editing available.</td>
</tr>
<tr>
<td>S, SP, and SS editing are not available.</td>
<td>S, SP, and SS editing available.</td>
</tr>
<tr>
<td>V editing available.</td>
<td>V editing not available.</td>
</tr>
<tr>
<td>= edit descriptor available.</td>
<td>= edit descriptor is not available. (However, a function subroutine which provides some of the features of this descriptor is available from Gary Villere of Informatics General Corp. upon request.)</td>
</tr>
</tbody>
</table>
4.0 SYSTEM UTILITIES

Differences between the CDC and CRAY versions of the following system utilities are compared.

**CDC**

**CALL DATE(idate)**

idate is returned as
10Hbmm/dd/yyb, where b denotes a blank

**CALL JDATE(idate)**

idate is returned as
5Ryyddd

**CALL SECOND(cputim)**

cputim is returned as a real number accurate to two decimal places

**CALL TIME(itime)**

itime is returned as
10Hbh:mm:ss.b, where b denotes a blank

**CALL SYSTEM(ierrno,msg)**

 ierrno = error number
 msg = execution time error
 message issued by user (Hollerith constant)

**CALL REMARK(msg)**

 msg = message issued by user to be placed in dayfile
 (Hollerith constant <= 9 10-character words)

**CRAY**

**CALL DATE(idate)**

idate is returned as
8Hmm:dd:yy

**CALL JDATE(idate)**

idate is returned as
5Hyyddd

**CALL SECOND(cputim)**

cputim is returned as a real number

**CALL TIME(itime)**

itime is returned as
8Hhh:mm:ss

**CALL CLOCK(itime)**

itime is returned as
8H:mm:ss

**CALL ABORT(msg)**

 msg = execution time error
 message issued by user
 (Hollerith constant <= 8 8-character words)

**CALL REMARK(msg)**

 msg = message issued by user to be placed in logfile
 (Hollerith constant <= 8 8-character words)
M=MEM(3HSCM) or M=MEM(3HLCM)

M is returned as the current field length

COS 1.11 Version:

INTEGER WC,T,DEL
DATA WC,T,DEL,DEL /5*0/
CALL MEMORY(WC,T,DEL,DEL)

WC is returned as the current field length

COS 1.12 Version:

INTEGER WC
CALL MEMORY('CURFL',WC)

WC is returned as the current field length

M=REQMEM(3HSCM,MREQ) or
M=reqMEM(3HSCM,MREQ)

MREQ = total field length requested by the user
M is returned as the new field length received

COS 1.11 Version:

INTEGER WC,T,DEL
DATA WC,DEL,DEL /3*0/
T = 1
WC = total field length desired
CALL MEMORY(WC,T,DEL,DEL)

L will be set if the job has received the maximum amount of memory allowed;
An error will occur if more memory is requested than is allowed.

COS 1.12 Version:

INTEGER WC
WC = total field length desired
CALL MEMORY('FL',WC)

This utility is only available from AMESLIB on the CDC 7600. It is accessed by the following statements on the JCL:

ACCESS,A,AMESLIB,ID=AMESLIB.
LIBRARY,A,*. 

For a further description of this utility, see "Computational Division User's Bulletin No. 214" available from the Central Computational Division Document Center.
Section 5

MASS STORAGE UTILITIES

5.0 MASS STORAGE UTILITIES

The mass storage INPUT/OUTPUT utilities discussed in this section are OPENKS, CLOSMS, READMS, WRITMS, and STINDX. On the CDC 7600, these utilities are supplied by the system. On the CRAY, however, these utilities must be accessed through AMESLIB as follows:

ACCESS,DN=READMS,PDN=READMS,ID=AMESLIB.
LDR,LIB=READMS.

Here the default dataset name of $BID is used, so the DN keyword is not specified. If the user specifies a dataset name different than $BID using the B keyword on the CFT control statement, then he must specify that dataset name on the LDR statement using the DN keyword (LDR,DN=dataset,LIB=READMS.).

Basically, these routines are used in the same manner on both machines. However, two notable differences have been found in the usage of OPENMS. Discussions on these differences follow.

<table>
<thead>
<tr>
<th>CDC</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPENMS allows the user to use a two-dimensional array for the defining the Master Index. It also allows the user to directly specify a starting location for this array in the call to OPENMS.</td>
<td>OPENMS does not allow the master Index array to be two-dimensional, nor does it allow a starting location to be directly specified in the call to OPENMS.</td>
</tr>
</tbody>
</table>

Example 5-1.

```
COMMON /INDEX/ INDEX(2,3),N

CALL OPENMS(LUN,INDEX(1,N), LENGTH,IT)
```

where LUN, LENGTH, and IT are previously defined.

Example 5-2. For example 5-1 to work on the CRAY do the following:

```
COMMON /INDEX/ INDX1(2),INDX2(2), INDX3(2),N

IF (N.DQ.1) CALL OPENMS(LUN,INDX1, LENGTH,IT)
IF (N.EQ.2) CALL OPENMS(LUN,INDX2, LENGTH,IT)
IF (N.EQ.3) CALL OPENMS(LUN,INDX3, LENGTH,IT)
```

where LUN, LENGTH, and IT are previously defined.
Section 6
CONVERSION HAZARDS

6.0 CONVERSION HAZARDS

6.1 Subroutines - Agreement in Number of Arguments

On the CDC 7600, if a user passes too many arguments to a subroutine the extra arguments are ignored. This is because the compiler reserves address space in the CALLING program for the table containing the addresses of those arguments.

On the CRAY, however, the compiler reserves the space for the argument address table in the routine being CALLED. Therefore, the amount of space reserved is only enough for the number of formal parameters specified in the subroutine declaration. It is the CALLING routine, however, which fills up this table at execution time. Thus, if a routine is called with too many arguments, the addresses of the extra arguments will over-write the lower portion of the routine immediately preceding the CALLED subroutine. This type of error is very difficult to trace as it often causes a section of code totally unrelated to the called subroutine to fail at execution time.
APPENDIX A

DIFFERENCES IN CDC AND CRAY

DEFAULT FILE NOMENCLATURE

<table>
<thead>
<tr>
<th>CDC</th>
<th>CRAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCL filenames:</td>
<td></td>
</tr>
<tr>
<td>INPUT</td>
<td>$IN</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>$OUT</td>
</tr>
<tr>
<td>OLDPL</td>
<td>$PL</td>
</tr>
<tr>
<td>NEWPL</td>
<td>$NPL</td>
</tr>
<tr>
<td>COMPILE</td>
<td>$CPL</td>
</tr>
<tr>
<td>LGO</td>
<td>$BLD</td>
</tr>
</tbody>
</table>

VAX <eor>,<eof> separators:

| ZZEOR     | /EOF      |
| ZZEOF     | /EOF      |