HOLLOMAN AIR FORCE BASE

CDC 6600 TO HP 3000 FORTRAN CONVERSION GUIDE

ROBERT G. TANTZEN

JANUARY 1974

CENTRAL INERTIAL GUIDANCE TEST FACILITY
6585TH TEST GROUP
HOLLOMAN AIR FORCE BASE, NEW MEXICO

CDC 6600 TO HP 3000 FORTRAN CONVERSION GUIDE

This paper contains the information necessary to convert Fortran programs from the CDC 6600 to the HP 3000. A separate paper will describe how most of the necessary changes can be made on the HP 3000 using the TEXT EDITOR program.

CONTENTS

A. Program Size

The total memory required to execute a program consists of four parts:

1. The program code, including library routines.

2. System routines for control of job execution.

3. Data area.

4. Buffer area for input and output.

On the 6600 almost all of this is contained in the user's field length. On the HP 3000, programs are divided into segments, only one of which needs to be in core memory at any one time. Library and system routines also can be shared by many users. To design an efficient program the user has only to consider his program and his own subroutines, and his data and buffer space.

A large program should be divided into code segments, each segment being one or more subprograms. Although the segment size is probably not very critical, a desirable size to aim for is 4000 to 8000 words, roughly about 600 lines of Fortran source code.

B. Data Storage Size

Both the data area (data stack) and the data buffers occupy space in core memory when a job executes. The data stack cannot be broken into segments, as is possible with programs. Therefore, it is very important to keep its size as small as possible.

The absolute machine limit for a data stack is 32000 words. The limit for any one program should be considerably smaller to allow multi-programming. We recommend to stay under 8000 words (4000 REAL values). The input/output buffer size depends on the size of the logical records to be transmitted. Buffering can be avoided altogether; this is done through control cards, so it is not a factor when writing the program.

The limits given above are initial recommendations. Actual experience with the HP 3000 will indicate any necessary changes.

C. Language Differences

Only those differences relevant to conversion of CDC 6600 programs are considered here.

1. Coding Procedures

a. Original card decks may have been punched on IBM 026 or 029 keypunch. If 026, then the original characters + = ( ) appear as $ # % < and must be changed to their original.

b. Comment Cards: If the character in column 1 is $ or *, change it to C.

c. Multiple statements per line, separated by $, are not allowed in HP. Each statement must start on a new card or line.

2. Constants (See also Paragraph F)

a. Octal CDC: $\pm$On$_1$....n$_i$ i = 6-20

$\pm$n$_1$....n$_i$B i = 1-20

HP: $\pm$%n$_1$....n$_i$ i = 1-6

b. Logical CDC: .TRUE. or .T.; .FALSE. or .F.

HP: .TRUE. .FALSE.

c. Hollerith

Hollerith constants, of form nH..... are allowed in DATA and FORMAT statements only. In assignment statements character strings enclosed in quotes must be used, "....". If in CDC the variable is type integer or logical, make it type CHARACTER*10 or less than 10 depending on how many characters are actually used.

In HP variables of other types can be assigned to character bit patterns as follows:

L = %"AB"L (logical, 1 word)

I = %"AB" (integer, 1 word)

71

R = %"ABCD"R (real, 2 words)

D = %"ABCDEF"D (double, 3 words)

d. Character string in FORMAT statement:

CDC *.....*

HP "....." or '.....'

e. Integer

In CDC an integer is often used to hold character information, typically read in under A10 format. In HP the variable should be made type CHARACTER*10; or an integer array, dimension 5, to be read into with 5A2 format.

3. Variables

A variable is considered logically .FALSE. if

CDC: whole word is zero
HP: least significant bit is zero

4. Expressions

a. Relational expressions

A op B, where op can be .EQ. .NE. .GT. .GE. .LT. .LE.

CDC: A and B can be of different type.
HP: Only integer, real, double precision can be mixed, but not complex or character. For the latter two, only .EQ. and .NE. may be used.

b. Logical Expressions

A op B op C .... where op can be .NOT. .AND. .OR. .XOR.
CDC: Does not have .XOR.
The others can be abbreviated to .N. .A. .O.
A and B can be type logical or integer
HP: No abbreviated notation
A and B must be type logical. For an integer variable I this can be achieved by using BOOL(I) intead of I.

c. Masking expressions (bit-by-bit logical operation)

A op B, where op can be .NOT. .AND. .OR. .XOR.
CDC: Does not have .XOR.
A and B any type except logical. Short forms .N. .A. .O. allowed
HP: A and B must be type logical. No short forms allowed.

If operands are type integer, use BOOL(I) to make them type logical. If more than 16 bits are involved "EQUIVALENCING" a LOGICAL array to the variable and performing the masking on a word by word basis. Sometimes using "partial word designators" may come in handy.

5. Replacement (Assignment) Statement

a. Mixed mode.

A = E (A is a variable, E is an expression). Normally, if A and E are of different type, an implicit type conversion occurs across the equal sign. Exceptions are:

CDC: If A is integer, E is logical, no conversion occurs.
HP: Only types INTEGER, REAL, DOUBLE PRECISION, COMPLEX may be mixed.

For type LOGICAL and CHARACTER, A and E must agree in type.

b. Multiple Assignment Statement

CDC: A = B = C..... = E
HP: Not allowed.

6. Type Declaration, Storage Allocation

a. CDC: DOUBLE PRECISION or just DOUBLE. The word TYPE may may precede any type name.
HP: DOUBLE PRECISION only. The word TYPE not allowed.

b. COMMON

CDC: Identifier can be numeric
HP: Identifier must start with a letter.

c. DATA

CDC: DATA can preset any variables and arrays except blank COMMON.
HP: DATA cannot preset any COMMON. Must use BLOCKDATA subprogram to do this.

CDC: Implied DO-loop notation in variable list is permitted, e.g. (A(J), J = 10, 20).
HP: Not permitted. The alternate form DATA ($V = C_1,....C_M$) must be changed to the normal form with slashes, DATA $V/C_1....C_M/$

7. Control Statements

a. Assigned GO TO

GO TO i ($M_1,.....M_M$)

CDC has the comma after the i optional.
HP requires the comma.

b. Computed GO

GO TO (M_1....M_M),I

CDC has the comma before the i optional.
HP requires the comma.

c. Two-way logical IF

IF (EXPR)M_1,M_2

HP does not have this. In CDC this can be used in two ways:

(1) EXPR is arithmetic (contains no relational operators like .GT.). Then EXPR is tested for zero (M_1) or non-zero (M_2). In this case use IF(EXPR)M_1, M_2, M_1.

(2) EXPR is logical, i.e., contains relational operators, or the name of a logical variable. Then use IF(INT(EXPR))M_1, M_2, M_1.

d. PAUSE and STOP

The optional number after a PAUSE or STOP is considered octal by CDC, decimal by HP. No change necessary.

e. RETURN

CDC: RETURN in a main program causes exit to the operating system.

HP: Undefined or not permitted ?

f. EXIT

CDC: CALL EXIT returns control to monitor.
HP: Use STOP.

8. Programs, Subprograms.

a. The CDC PROGRAM card contains all filenames used. This must be deleted.

b. CDC may have FORTRAN VI, FORTRAN IV, etc in front of PROGRAM, SUBROUTINE, or FUNCTION. Remove that. (FORTRAN VI forces pre-testing of DO-loops and allows END to act as RETURN in subroutines.)

c. ENTRY statement in subroutines.

CDC provides alternate entry points to subroutines.

HP does not. Add one parameter to subroutine and use it there to jump to the desired point.

9. Segmentation, Overlays.

CDC: Used to save memory space.
HP: Due to the different design architecture of the 3000 segmentation is performed differently. Remove all OVERLAY and CALL OVERLAY cards.

10. Input, Output

a. Standard Input, Standard List, Standard Punch.

CDC:		HP:	
	READ fm, List		READ (5,fm) List
	PRINT fm, List		WRITE (6,fm) List
	PUNCH fm, List		WRITE (4,fm) List

b. NAMELIST

HP does not have this feature. Replace with regular formatted READ or WRITE.

c. BUFFER IN, BUFFER OUT, IF UNIT, LENGTH.

HP does not have this, replace with

READ (i,ERR=M_1, END = M_2) List
WRITE (i,ERR = M)

Keep in mind that BUFFER transfers data directly to or from core, while READ and WRITE normally use buffers. Buffing can be avoided by specifying NOBUF in FILE command. When reading a tape record with parity errors it is mandatory that the program can examine the record. How this can be achieved has not yet been resolved.

CDC allows to BUFFER IN a logical record of unknown length, then use the LENGTH function to find the number of words actually read. To accomplish this in HP Fortran, we may have to call FREAD directly from Fortran or via an SPL procedure.

d. End-of-file check after READ.

CDC has IF(EOF,i)
IF(ENDFILEi)
HP does this with
READ (i, END=M)

e. Special forms of READ and WRITE

CDC:		HP:	
	READ INPUT TAPE i, fm, List		READ (i,fm) List
	WRITE OUTPUT TAPE i, fm, List		WRITE (i,fm) List
	READ TAPE i, List		READ (i) List
	WRITE TAPE i, List		WRITE (i) List

f. ENDFILE 1

CDC: Writes EOF
HP: Writes EOF and closes the file. In order to write multi-file tapes, it must be possible to write an EOF without invoking file closure. (Under investigation.)

g. ENCODE, DECODE

```
CDC:  DECODE (n,fm,CA) list   [CA = integer array]
HP:   READ (CA,fm) list       [CA = character variable, size n]
Same for ENCODE, WRITE.
```

D. Library Function.

1. HP does not have ASIN(X) and ACOS(X). Both can be expressed in terms of ATAN. If used frequently the two functions can be written and placed in the library.

2. Randon Numbers

CDC has RANF(R) as a library function.

HP has RAND as an external subroutine. User should check if the numbers generated meet his criteria.

3. Shift Function

CDC has LS(A,i) to shift one word left or right.

HP has K=K(b,n), partial-word designator, which can be used for right shift. Some re-programming may be necessary because the difference in word length enters into the picture.

4. Byte Handling

CDC has the INBY and ENBY functions to manipulate bytes of arbitrary length.

Possibly an SPL procedure should be written, or reprogramming using partial word designators solves the problem.

E. Library Subroutines

1. Time and Date

CDC: DATE gives day, month, year
CLOCK gives hours, min, sec

HP: CHRONOS gives all of above, but in a different format. Probably must write a Fortran callable subroutine which in turn calls the CHRONOS intrinsic (no parameters).

CDC: SECOND gives CP time used
TIMTGØ gives CP time remaining
HP: PROCTIME gives CP time used. Remaining time not obtainable. Probably must write subroutine since PROCTIME has no parameters and different format for time returned.

2. File Handling

The following CDC Fortran subroutines seem not to have an HP equivalent:

PARTOUT moves a file to the appropriate output queue (print, punch, microfilm) and rewinds that local file to accept further information.

SKFILE skips files forward or backward on multi-files.

RETURNS releases a tape file and the tape drive.

UNLOADS releases a tape file, not the tape drive.

The last two routines allow the return of files before the end of a program.

IDENT reads or writes ID-records on data files. This subroutine will be programmed and placed on the system library.

3. Messages

CDC routines REMARK, DISPLA send a message to the operator or the dayfile.

HP has PRINTOP for message to operator. The dayfile information is mixed with other putput on $STDLIST.

F. Effects of Different Word Size

Precision and Range of Numbers

	CDC	HP
Type REAL, significant decimals	14.5	6.9
smallest absolute value	10^{-294}	10^{-77}
largest absolute value	10^{322}	10^{77}
Type DOUBLE, significant decimals	29	11.7
smallest absolute value	10^{-294}	10^{-77}
largest absolute value	10^{322}	10^{77}
Type integer, largest value	$2^{48}-1$	$2^{15}-1$
Integer subscript, largest value	$2^{15}-1$	$2^{15}-1$

Points to Consider:

1. Check if any integer variables or constants may be greater than $2^{15}-1 = 32767_{10}$.

2. Check if any real variables should be made double precision.

3. Check if any real or double precision variables could exceed 10^{77}.

4. Check if CDC program has any double precision variables or arrays. If so, analyze the problem to see if it can be done at all on the HP 3000.

5. Check if integer variables or arrays are used to store character information, 10 characters per word. If so, may have to use type CHARACTER.